

# IBM Db2 13 for z/OS Performance Topics

Neena Cherian  
Nguyen Dao  
Reynaldo Fajardo  
Akiko Hoshikawa  
Peng Huang  
Maggie Ou Jin  
Ping Liang  
Jie Ling  
Todd Munk  
Bart Steegmans

Jasmi Thekveli  
Lingyun Wang  
Chung Wu  
Chongchen Xu  
Huiya Zhang  
Xiao Wei Zhang  
Xue Lian Zhang



 Analytics

Data and AI





IBM Redbooks

**IBM Db2 13 for z/OS Performance Topics**

January 2023

**Note:** Before using this information and the product it supports, read the information in “Notices” on page xi.

**First Edition (January 2023)**

This edition applies to IBM Db2 13 for z/OS.

**© Copyright International Business Machines Corporation 2023. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.



# Contents

<b>Notices</b> .....	xi
Trademarks .....	xii
<b>Preface</b> .....	xiii
Authors .....	xiv
Now you can become a published author, too! .....	xvi
Comments welcome .....	xvii
Stay connected to IBM Redbooks .....	xvii
<b>Chapter 1. Introduction</b> .....	1
1.1 Overview of Db2 13 .....	2
1.1.1 Key Db2 13 features .....	3
1.2 High-level performance expectations .....	6
1.3 Synergy with the IBM Z platform .....	7
1.3.1 IBM zSystems hardware leverage .....	7
1.3.2 Synergy with the IBM z/OS operating system .....	9
1.4 Db2 for z/OS and its ecosystem performance .....	9
1.4.1 IBM Db2 Analytics Accelerator and IBM Integrated Synchronization .....	10
1.4.2 IBM Db2 AI for z/OS .....	10
1.5 How to use this book .....	10
<b>Chapter 2. Scalability and availability</b> .....	13
2.1 Below-the-bar local agent storage reduction .....	14
2.1.1 Performance measurement .....	14
2.1.2 Conclusion .....	15
2.2 Extended common service area storage reduction .....	15
2.2.1 Performance measurement .....	16
2.2.2 Conclusion .....	16
2.3 Storage manager contraction optimization in Db2 13 .....	16
2.3.1 Requirements .....	17
2.3.2 Performance measurement .....	17
2.4 Raising the DS MAX limit .....	21
2.4.1 Requirements .....	21
2.4.2 Performance measurements .....	22
2.4.3 Monitoring information .....	25
2.4.4 Usage considerations .....	25
2.4.5 Conclusion .....	25
2.5 Reduced security manager contention for Db2 access .....	26
2.5.1 Enabling plan authorization cache .....	26
2.5.2 Performance measurement .....	27
2.6 Fast index traversal enhancements .....	27
2.6.1 Performance measurements for FTB support for non-unique indexes .....	28
2.6.2 Performance measurements for an FTB key size increase .....	32
2.6.3 Usage and monitoring .....	36
2.6.4 Conclusion .....	39
2.7 DDF improvements .....	39
2.7.1 Changing DDF communication buffers to use DBAT thread storage .....	39
2.7.2 Gradual DBAT contraction .....	40

<b>Chapter 3. Synergy with the IBM Z platform</b> .....	43
3.1 IBM z16 .....	44
3.1.1 Db2 workload performance with IBM z16 .....	44
3.2 Data-sharing overhead reduction with short reach coupling links .....	46
3.2.1 Requirements .....	46
3.2.2 Performance measurement .....	46
3.2.3 Monitoring information .....	47
3.2.4 Conclusion .....	48
3.3 CFLEVEL 25 and Db2 CF structures .....	48
3.3.1 Measurement environment .....	48
3.3.2 Results .....	48
3.4 GBP residency time .....	51
3.4.1 Performance measurement .....	52
3.5 System Recovery Boost: IBM z15 and later .....	60
3.5.1 Requirements .....	60
3.5.2 Performance measurement .....	60
3.5.3 Setup and monitoring information .....	67
3.5.4 Usage considerations .....	68
3.5.5 Conclusion .....	68
3.6 REORG table space with variable length records on IBM z15 with IBM Z Sort .....	69
3.6.1 Requirements .....	69
3.6.2 Performance measurement .....	69
3.6.3 Usage considerations .....	70
3.6.4 Conclusion .....	71
3.7 Leveraging Z Sort by SQL sort .....	71
3.7.1 Requirements .....	71
3.7.2 Performance measurement .....	71
3.7.3 Monitoring information .....	73
3.7.4 Usage considerations .....	74
3.8 Large object compression .....	75
3.8.1 Performance measurement .....	75
3.8.2 Conclusion .....	79
3.9 Asynchronous cross-invalidation: IBM z14 and later .....	80
3.9.1 Requirements .....	80
3.9.2 Performance measurement .....	80
3.9.3 Monitoring information .....	82
3.9.4 Usage considerations .....	82
3.9.5 Conclusion .....	82
3.10 zHyperLink I/O .....	82
3.10.1 How to enable zHyperLink I/O for Db2 .....	84
3.10.2 zHyperLink and database read I/O .....	86
3.10.3 Requirements .....	89
3.10.4 Performance measurements .....	89
3.10.5 Monitoring Db2 zHyperLink database I/O .....	95
3.10.6 zHyperLink read usage considerations .....	100
3.10.7 zHyperLink read conclusion .....	102
3.10.8 zHyperLink and writes .....	102
3.10.9 Requirements .....	103
3.10.10 Performance measurements .....	103
3.10.11 Monitoring zHyperLink write .....	109
3.10.12 zHyperLink write usage considerations .....	111
3.10.13 zHyperLink write conclusion .....	111
3.11 Huffman compression for data .....	112

3.11.1	Enabling Huffman compression . . . . .	112
3.11.2	Space savings from Huffman compression. . . . .	114
3.11.3	Huffman compression performance . . . . .	116
3.11.4	Performance considerations . . . . .	124
3.11.5	Using DSN1COMP for Huffman compression estimation. . . . .	125
3.12	Encrypting Db2 data with z/OS data set encryption and CF structure encryption. . .	127
3.12.1	Requirements . . . . .	128
3.12.2	Performance measurements for data set encryption . . . . .	128
3.12.3	Performance measurements for the REORG utility and encryption . . . . .	138
3.12.4	Performance measurements for GBP CF structure encryption. . . . .	141
3.12.5	Performance measurements for IBM brokerage transaction workload. . . . .	145
3.12.6	Monitoring information . . . . .	148
3.12.7	Usage considerations . . . . .	148
3.12.8	Conclusion . . . . .	148
<b>Chapter 4.</b>	<b>Workload-level measurements . . . . .</b>	<b>149</b>
4.1	Performance regression bucket . . . . .	150
4.1.1	Performance measurement description . . . . .	150
4.1.2	Performance results . . . . .	150
4.1.3	Conclusion . . . . .	153
4.2	IBM brokerage transaction workload (version 1). . . . .	153
4.2.1	Two-way and one-way data-sharing and non-data-sharing performance measurements . . . . .	153
4.2.2	Performance measurement. . . . .	153
4.2.3	Conclusion . . . . .	156
4.3	IBM brokerage transaction workload (version 2): Effect of REBIND (one-way data sharing) on performance . . . . .	156
4.3.1	Requirements . . . . .	156
4.3.2	Performance measurements. . . . .	157
4.3.3	Performance results . . . . .	157
4.3.4	Conclusion . . . . .	160
4.4	Distributed IBM Relational Warehouse Workload workload . . . . .	160
4.4.1	Distributed IBM Relational Warehouse Workload CPU and ITR comparison. . .	161
4.5	High-insert batch workloads . . . . .	162
4.5.1	Performance measurement environment . . . . .	162
4.5.2	Performance results . . . . .	162
4.6	Relational transaction workload (CICS and Db2) . . . . .	177
4.6.1	Performance measurement. . . . .	177
4.7	IBM query regression workload. . . . .	180
4.7.1	Characteristics of query workloads and testing environment . . . . .	181
4.7.2	Performance results . . . . .	182
4.8	SAP banking workloads . . . . .	183
4.8.1	Scalability test environment . . . . .	183
4.8.2	Scalability performance results . . . . .	183
4.8.3	Data-sharing test environment . . . . .	185
4.8.4	Data-sharing test results. . . . .	185
4.8.5	Conclusion . . . . .	186
<b>Chapter 5.</b>	<b>Data sharing . . . . .</b>	<b>187</b>
5.1	Partition-by-range table space relative page numbering enhancements . . . . .	188
5.1.1	Requirements . . . . .	188
5.1.2	Performance measurement. . . . .	188
5.1.3	Usage considerations . . . . .	193

5.1.4 Conclusion .....	193
5.2 Group buffer pool cast-out-related changes in Db2 13 .....	194
5.2.1 Requirements .....	194
5.2.2 Performance measurements .....	194
5.3 Data sharing with distance .....	197
5.3.1 Requirements .....	197
5.3.2 Performance measurement .....	197
5.3.3 Monitoring information .....	209
5.3.4 Conclusion .....	209
<b>Chapter 6. SQL Data Insights</b> .....	<b>211</b>
6.1 Enabling AI query .....	212
6.1.1 Requirements .....	212
6.1.2 Performance measurement .....	213
6.1.3 Monitoring information .....	225
6.1.4 Usage considerations .....	226
6.1.5 Conclusion .....	228
6.2 Running an AI query .....	228
6.2.1 Requirements .....	229
6.2.2 Performance measurement .....	229
6.2.3 Monitoring information .....	238
6.2.4 Usage considerations .....	238
6.2.5 Conclusion .....	239
6.3 Summary .....	239
<b>Chapter 7. Application concurrency</b> .....	<b>241</b>
7.1 Inserting improvements into partition-by-growth table spaces .....	242
7.1.1 Requirements .....	242
7.1.2 Retrying a search of previously failed partitions .....	242
7.1.3 Smarter cross-partition search .....	247
7.1.4 Conclusion .....	251
7.2 Index look-aside optimization .....	251
7.2.1 Requirements .....	252
7.2.2 Performance measurements .....	252
7.3 Index management diagnostic and serviceability enhancements .....	261
7.3.1 The new index split fields in SYSIBM.SYSINDEXSPACESTATS .....	261
7.3.2 The new IFCID 396 .....	262
7.3.3 Diagnosis of index split problems .....	263
7.4 Application timeout and deadlock control .....	265
7.4.1 Performance measurements .....	265
7.4.2 Monitoring information .....	265
7.4.3 Usage considerations .....	266
7.5 Enhanced support for profile tables .....	266
7.5.1 Performance measurements .....	267
7.5.2 Usage considerations .....	270
7.5.3 Conclusion .....	270
<b>Chapter 8. Query performance</b> .....	<b>271</b>
8.1 Sort optimization for Db2 built-in functions with multiple DISTINCT, GROUPING SETS, and PERCENTILE clauses .....	272
8.1.1 Requirements .....	272
8.1.2 Performance measurements .....	272
8.1.3 Conclusion .....	273
8.2 SUBSTR support for the LISTAGG function .....	273

8.2.1	Requirements	273
8.2.2	Performance measurements	273
8.2.3	Conclusion	274
8.3	Improving ORDER BY sorts for long VARCHAR columns	274
8.3.1	Requirements	274
8.3.2	Performance measurements	275
8.3.3	Monitoring information	276
8.4	APREUSE storage reduction	276
8.4.1	Requirements	277
8.4.2	Performance measurements	277
8.4.3	Conclusion	278
<b>Chapter 9. IBM Db2 for z/OS utilities</b>		279
9.1	LOAD PRESORT	280
9.1.1	Requirements	280
9.1.2	Performance measurements	280
9.1.3	Usage considerations	282
9.1.4	Conclusion	282
9.2	Online conversion from partition-by-growth to PBR	282
9.2.1	Requirements	282
9.2.2	Performance measurements	283
9.2.3	Usage considerations	284
9.2.4	Conclusion	285
9.3	Page sampling support for inline statistics	285
9.3.1	Requirements	286
9.3.2	Performance measurements	286
9.3.3	Usage considerations	289
9.3.4	Conclusion	290
9.4	Utility history	290
9.4.1	Requirements	291
9.4.2	Performance measurements	291
9.4.3	Usage considerations	292
9.4.4	Conclusion	292
9.5	Redirected recovery	292
9.5.1	Requirements	293
9.5.2	Performance measurements	293
9.5.3	Conclusion	299
9.6	Migrating a multi-table table space to PBG universal table spaces	300
9.6.1	Requirements	300
9.6.2	Performance measurement	300
9.6.3	Usage considerations	302
9.6.4	Conclusion	302
9.7	REORG INDEX NOSYSUT1	303
9.7.1	Requirements	303
9.7.2	Performance measurements	303
9.7.3	Monitoring information	305
9.7.4	Usage considerations	305
9.7.5	Conclusion	305
<b>Chapter 10. Performance enhancements for IDAA for z/OS and IBM Db2 for z/OS Data Gate</b>		307
10.1	IBM Integrated Synchronization	308
10.1.1	Requirements	308
10.1.2	Performance measurement	308

10.1.3	Monitoring information	311
10.2	Cluster load	311
10.2.1	Requirements	312
10.2.2	Performance measurement	312
10.2.3	Usage considerations	313
10.2.4	Conclusion	313
10.3	COLJOIN 1 enablement	314
10.3.1	Requirements	314
10.3.2	Performance measurement	314
10.3.3	Usage considerations	315
10.4	Direct I/O for IBM Db2 Analytics Accelerator on IBM Z	315
10.4.1	Requirements	316
10.4.2	Performance observations	316
10.5	Automatic statistics collection and collection improvement	316
10.5.1	Early statistics collection	316
10.5.2	Copy Table Statistics	317
10.5.3	Explicit RUNSTATS	319
10.5.4	Summary	321
10.6	Query workload comparison between Db2 12 and Db2 13 for z/OS	321
10.6.1	Performance measurement	321
10.7	Performance impact of AT-TLS enablement	322
10.7.1	AT-TLS configuration for IBM Integrated Analytics Accelerator	322
10.7.2	Results and usage considerations	322
10.8	Performance overview for IBM Db2 for z/OS Data Gate	323
10.8.1	IBM Db2 for z/OS Data Gate deployment options	323
10.8.2	Performance test strategies and results	325
10.8.3	Performance measurement: Load throughput and synchronization latency	325
10.8.4	Performance measurement: Query	328
10.8.5	Conclusion	329
<b>Chapter 11</b>	<b>Installation and migration</b>	<b>331</b>
11.1	Migration performance	332
11.1.1	Requirements	332
11.1.2	Performance measurements	333
11.1.3	Monitoring information	336
11.1.4	Usage considerations	337
11.1.5	Conclusion	337
11.2	Performance-related subsystem parameter changes	337
11.2.1	New subsystem parameters that are related to performance	337
11.2.2	Performance-related subsystem parameters that were updated	338
11.2.3	Obsolete subsystem parameters that are related to performance	341
<b>Chapter 12</b>	<b>Monitoring and instrumentation</b>	<b>343</b>
12.1	New IFCID 396 to track index page split activity	344
12.2	Enhanced distributed thread monitoring	344
12.2.1	New IFCID 411 for recording DDF application statistics	344
12.2.2	New IFCID 412 for recording DDF client user ID statistics	346
12.2.3	New fields in IFCID 365 for recording DDF location statistics	348
12.2.4	New fields in IFCID 1 for recording global DDF activity	350
12.2.5	Db2 AI for z/OS 1.5 enhancements for DDF	351
12.3	IFCID 003: Recording the longest wait time for certain suspension types	352
12.4	Group buffer pool residency time enhancements	356
12.5	IFCID 369 enabled by default when STATISTICS CLASS(1) starts	356

12.6	IFCID changes for application timeout and deadlock control	357
12.7	Partition range support for IFCID 306 READS requests	358
12.8	New fields in IFCID 3 for SQL Data Insights	360
12.9	New fields in IFCID 2 for plan authorization cache	360
12.10	New field in IFCID 389 for fast index traversal	361
12.11	IFCID changes that are related to storage manager large object contraction	361
12.12	IFCID changes that are related to latch-level expansion	362
12.13	Obtaining the most current information about IFCID changes	363
12.14	Summary	363
<b>Chapter 13. IBM Db2 AI for z/OS benefits and capacity requirement</b>		<b>365</b>
13.1	How Db2ZAI helps reduce application cost: SQL optimization	366
13.2	How Db2ZAI helps with Db2 subsystem management: System assessment and performance insights	371
13.2.1	System assessment and performance insights use case example	372
13.2.2	System assessment in-depth recommendations	375
13.3	How Db2ZAI helps manage inbound network traffic: DCC	376
13.3.1	Db2ZAI DCC use case: Setting up profiles to monitor and control distributed connections	377
13.3.2	Db2ZAI 1.5 DCC use case: Visualizing distributed workload activities with the dashboard	380
13.4	Summary of capacity requirements and general recommendations	385
13.5	Detailed capacity evaluations for Db2ZAI 1.5	387
13.5.1	High-level capacity evaluation results	390
13.5.2	Detailed capacity evaluation results	393
13.6	Monitoring information	401
<b>Appendix A. IBM Db2 workloads</b>		<b>403</b>
	IBM Relational Warehouse Workload	404
	Relational transaction workload	405
	IRWW distributed workload	406
	IBM brokerage transaction workload	406
	Sample performance regression workload	407
<b>Appendix B. Artificial intelligence semantic queries</b>		<b>409</b>
	SQL Data Insights AI semantic queries that are used in scenario 1 and scenario 2	410
	AI semantic queries that use the AI_SIMILARITY function	410
	AI semantic queries that use the AI_SEMANTIC_CLUSTER function	413
	AI semantic queries that use the AI_ANALOGY function	415
	SQL Data Insights semantic queries that are used in scenario 3	416
	AI semantic query that uses the AI_SIMILARITY function	416
	AI semantic query that uses the AI_SEMANTIC_CLUSTER function	416
	AI semantic query that uses the AI_ANALOGY function	417
	SQL Data Insights semantic queries that are used in scenario 4	417
	AI semantic query that uses the AI_SIMILARITY function	418
	AI semantic query that uses the AI_SEMANTIC_CLUSTER function	418
	AI semantic query that uses the AI_ANALOGY function	419
<b>Appendix C. IBM Db2 Analytics Accelerator for z/OS workloads</b>		<b>421</b>
	TPC-H query workload	422
	Customer FD workload	422
<b>Abbreviations and acronyms</b>		<b>425</b>

<b>Related publications</b> .....	427
IBM Redbooks .....	427
Online resources .....	427
Help from IBM .....	428



# Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.


## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

AIX®	IBM Research®	Redbooks®
CICS®	IBM Spectrum®	Redbooks (logo)  ®
Cognos®	IBM Watson®	Tivoli®
Db2®	IBM Z®	WebSphere®
DS8000®	IBM z13®	z/OS®
FICON®	IBM z14®	z13®
GDPS®	IBM z16™	z15®
HyperSwap®	InfoSphere®	z16™
IBM®	Netezza®	zEnterprise®
IBM Cloud®	OMEGAMON®	
IBM Cloud Pak®	RACF®	

The following terms are trademarks of other companies:

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Red Hat and OpenShift are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

# Preface

IBM® Db2® 13 for z/OS delivers innovations that can improve your ability to make well-informed business decisions. As the industry's first relational database that integrates artificial intelligence (AI) into SQL queries, Db2 13 combines deep-learning capabilities with advanced IBM Z® technologies to reveal hidden relationships across tables and views in your Db2 data.

As with earlier releases of Db2, Db2 13 continues to enhance availability, scalability, security, and resiliency, and applies optimization to improve the performance of your operational processes. These improvements include the following ones:

- ▶ New business insights without complex AI application deployment
- ▶ Scalability and performance improvements through smarter optimization without tuning actions
- ▶ Highly available (HA) system management with greater resiliency and flexibility
- ▶ Greater insights into managing complex enterprise systems

Db2 13 is built on the continuous development delivery model that was introduced in Db2 12 for z/OS, and it continues to leverage a tight synergy between IBM zSystems hardware and IBM z/OS®.

This IBM Redbooks® publication provides performance comparisons between Db2 13 and Db2 12, including high-level performance expectations when you migrate to Db2 13. It also provides measurement results and usage considerations for using the new capabilities that are delivered in Db2 13. Because Db2 13 is built on Db2 12 and the functions that were delivered through the continuous delivery stream, measurements are provided for the key performance features that were delivered in Db2 12 and the Db2 ecosystem since *IBM Db2 12 for z/OS Performance Topics*, SG24-8404 was published in September 2017. These features include various IBM Z platform synergy items and new features that were added in IBM Db2 Analytics Accelerator (IDAA), IBM Db2 for z/OS Data Gate, and IBM Db2 AI for z/OS (Db2ZAI).

The performance measurements in this book were generated at the IBM Silicon Valley Laboratory under specific and tightly controlled conditions. Your results are likely to vary due to differences in your conditions and workloads.

For the purposes of this publication, it is assumed that you are familiar with the performance aspects of Db2 for z/OS. For more information about the functions that were delivered in Db2 13, see the [Db2 13 for /OS](#) and *IBM Db2 13 for z/OS and More*, SG24-8527.

# Authors

This book was produced by the IBM Db2 for z/OS performance team.

**Neena Cherian** is a member of the Db2 for z/OS Performance team at the IBM Silicon Valley Laboratory. She has been with IBM for over 30 years, with 14 years that have been devoted to Db2 for z/OS performance. Her areas of expertise include Online Transaction Processing (OLTP), direct access storage device (DASD) storage-related system performance, and SQL Data Insights.

**Nguyen Dao** is a member of the Db2 for z/OS performance team at the IBM Silicon Valley Laboratory. He has been with IBM for 22 years, 13 of which have been spent working on Db2 for z/OS performance. His areas of expertise include system-level and distributed environment performance.

**Reynaldo Fajardo** is a Software Engineer at the IBM Silicon Valley Laboratory. He has a year of experience in Db2 for z/OS performance. He holds a degree in Computer Science from the University of California, Santa Cruz. His areas of expertise include query performance and analysis for the IDAA.

**Akiko Hoshikawa** is a Distinguished Engineer in IBM Data and AI at the IBM Silicon Valley Laboratory. She is a Db2 for z/OS architect with overall responsibility for the performance and strategic integration of AI within Db2, which includes SQL Data Insights and many other optimizations in Db2. She does hands-on work with customers to provide performance evaluation, tuning, benchmarks, and to design performance features to solve customer challenges. Akiko has been involved with or written IBM Redbooks publications about Db2 for z/OS Performance Topics since 1998.

**Peng Huang** is a member of the Db2 for z/OS performance team at the IBM China Development Laboratory. She has been working for the IBM Db2 for z/OS team for 10 years, 7 of which have been focused on Db2 for z/OS performance in the area of query optimization.

**Maggie Ou Jin** is a member of the Db2 for z/OS performance team at the IBM Silicon Valley Laboratory. She has been with IBM for 20 years. Her areas of expertise include Db2 utilities, IDAA performance, and query optimization.

**Ping Liang** is a member of the Db2 for z/OS performance department at the IBM China Development Laboratory. He has been working for IBM Db2 for z/OS team for 16 years, with eight years that have been devoted to Db2 for z/OS performance. His areas of expertise include application optimization, system performance, and spring-boot application design and development.

**Jie Ling** is a member of the Db2 for z/OS performance team at the IBM China Development Laboratory. She has been with IBM for 14 years, 7 of which have been focused on Db2 for z/OS performance. Her areas of expertise include query optimization and high insert performance.

**Todd Munk** is a Senior Software Engineer at the IBM Silicon Valley Laboratory. He has been with IBM for 27 years with the Db2 for z/OS Performance team. His areas of expertise include distributed connectivity and stored procedures performance. He is spending much of his time on Scala back-end development for IBM Db2ZAI.

**Bart Steegmans** is a Consulting Db2 Product Support Specialist from IBM Belgium. He is working remotely for the Silicon Valley Laboratory in San Jose and providing technical support for Db2 for z/OS performance problems. He has over 30 years of experience in Db2. Before joining IBM in 1997, Bart worked as a Db2 system administrator at a banking and insurance group. His areas of expertise include Db2 performance, database administration, and backup and recovery. Bart is co-author of numerous IBM Redbooks publications about Db2 for z/OS.

**Jasmi Thekveli** is a member of the SAP on IBM Z performance team that is based in Poughkeepsie, NY. She has been with IBM for 20 years, with over 2 years in SAP on IBM Z performance, and 8 years with the Db2 for z/OS performance team. Her areas of expertise include SAP performance on Db2 for z/OS, IDAA performance, and query optimization.

**Lingyun Wang** is a member of the Db2 for z/OS performance team at the IBM Silicon Valley Laboratory. She has been with IBM for 18 years, 15 of which have been spent working on Db2 for z/OS performance. Her areas of expertise include query optimization, IDAA performance, and replication.

**Chung Wu** is a member of the Db2 for z/OS performance team at the IBM Silicon Valley Laboratory. He has been with IBM for more than 30 years, with over 20 years that have been devoted to Db2 for z/OS performance. His areas of expertise include system-level performance, and OLTP.

**Chongchen Xu** is a member of the Db2 for z/OS performance team at the IBM Silicon Valley Laboratory. He has been with IBM Db2 for z/OS for 17 years. His areas of expertise include Db2 performance regression, batch processing, and OLTP performance.

**Huiya Zhang** is a Performance Engineer at the IBM Silicon Valley Laboratory. Before joining the Db2 for z/OS performance development team, she had more than 10 years of experience working in the Db2 for z/OS Client Success team with a focus on Db2 Relational Data System (RDS).

**Xiao Wei Zhang** is a Senior Software Engineer at the IBM China Development Laboratory. He has over 15 years of experience with database development and performance at IBM.

**Xue Lian Zhang** is a member of the Db2 for z/OS performance team at the IBM China Development Laboratory. She started her mainframe career in the finance industry in 2007 and joined IBM in 2012. She has been working on Db2 for z/OS since 2016 and focuses on the OLTP areas.

Special thanks to the following people for their unique contributions to this publication and the overall project:

- ▶ Leilei Li and Huiya Zhang for leading teams across organizations and geographies to ensure the timely delivery of this publication.
- ▶ Akiko Hoshikawa and Bart Steegmans for their generous technical guidance throughout the entire writing and reviewing process.
- ▶ Eric Radzinski and Paul McWilliams for providing technical editing, writing guidance, and improving the format and style of this publication.
- ▶ Steven Brazil and Mo Townsend for their management support.

Thanks to the following people for their contributions to this project:

Marcela Adan, Bryan Davis, Wade Wallace  
**IBM Redbooks**

Gayathiri Chandran, Rick Chang, Julie Chen, Ying-lin Chen, Tammie Dang, Thomas Eng, Sarbinder Kallar, Tran Lam, Allan Lebovitz, John Lyle, Manvendra Mishra, Randy Nakagawa, Steven Ou, Sharon Roeder, Michael Shadduck, Tom Toomire, Frances Villafuerte, Bituin Vizconde, Sherry Yang  
**IBM Db2 for z/OS Development**

Craig Friske, Koshy John, Laura Kunioka-Weis, Patrick Malone, Ka-Chun Ng,  
**IBM Db2 for z/OS Utilities - Rocket Software**

Christian Michel  
**IBM Db2 Analytics Accelerator**

Pooja Bhargava, Brian Lee, Nicholas Marion, Jose Neves, Dale Riedy, Guo Ran Sun, Dave Surman  
**IBM zSystems Development**

Joe Gentile, Fimy Hu, Veng Ly  
**IBM zSystems Performance**

Rajesh Bordawekar  
**IBM Research®**

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an IBM Redbooks residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:  
[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an email to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, IBM Redbooks  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- ▶ Find us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- ▶ Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>







# Introduction

Scalability and performance continue to be important themes in IBM Db2 for z/OS and its ecosystem. Performance optimization in Db2 13 for z/OS is built on continuous delivery of Db2 12 for z/OS and includes in-memory optimization and autonomic learning behavior based on prior executions.

This chapter provides an overview of the key features of Db2 13 and describes the Db2 13 performance improvements and expectations (based on laboratory measurements) for different types of workloads when you migrate to Db2 13 from Db2 12. You can realize many performance benefits, including potential CPU or cost reductions, by migrating to Db2 13 and rebinding applications, but other enhancements might require extra actions or memory allocation to leverage them. A summary of IBM Z platform synergy and key updates in ecosystem products is also provided.

This chapter includes the following topics:

- ▶ Overview of Db2 13
- ▶ High-level performance expectations
- ▶ Synergy with the IBM Z platform
- ▶ Db2 for z/OS and its ecosystem performance
- ▶ How to use this book

## 1.1 Overview of Db2 13

IBM Db2 13 for z/OS delivers innovation to improve your business decisions through deep learning capabilities as the industry's first relational database that integrates artificial intelligence (AI) into Db2 SQL queries. This innovation, also known as *SQL Data Insights*, can be quickly adapted by existing Db2 applications to perform AI semantic query functions to improve real-time decisions, such as detecting fraudulent behavior or improving the recommendations that are based on existing similar behavior.

In addition to AI infusion, Db2 13 and its ecosystem continue to enhance availability, scalability, security, and resiliency, and applies optimization to improve the performance of your operational processes.

Db2 13 is built on continuous development and delivery that was introduced in Db2 12 for z/OS and continues to leverage a tight synergy with both IBM zSystems hardware and z/OS.

The Db2 ecosystem includes IBM zSystems hardware synergy and extra features and components. It includes IBM Db2 Analytics Accelerator (IDAA) and IBM Db2 for z/OS Data Gate, both of which are tightly integrated with Db2 for z/OS to support business-critical reporting and analytic workloads with speed. It also includes IBM Db2 AI for z/OS (Db2ZAI) to help improve the operational performance and maintain the health of Db2 systems by using AI capability.

The benefits that Db2 13 provides include the following ones:

- ▶ New business insights without complex AI application deployment
- ▶ Scalability and performance improvements through smarter optimization without the need for extra tuning actions
- ▶ Highly available (HA) system management with greater resiliency and flexibility
- ▶ Greater insights in managing complex enterprise systems

This publication provides performance comparisons of Db2 13 against Db2 12 for various types of workloads when you migrate to Db2 13. It also provides measurement results for the new features and considerations for using these features.

Because Db2 13 is built on Db2 12 continuous delivery, it also includes measurements for the key performance features that were delivered in the Db2 12 continuous delivery stream and the Db2 ecosystem since *IBM Db2 12 for z/OS Performance Topics*, SG24-8404 was published in September 2017. These features include IBM Z platform synergy items, such as the following ones:

- ▶ The leveraging of group buffer pool (GBP) residency time in IBM z16™
- ▶ Coupling Facility (CF) scalability and coupling short link improvement in IBM z16
- ▶ System Recovery Boost (SRB) in IBM z15® and later
- ▶ IBM Integrated Accelerator for IBM Z Sort (Z Sort) in IBM z15 and later
- ▶ Large object (LOB) compression improvement in IBM z15 and later
- ▶ Huffman compression in IBM z14® and later
- ▶ Asynchronous cross-invalidation in IBM z14 and later
- ▶ IBM zHyperLink in IBM z14 and later
- ▶ Transparent data set encryption improvement in IBM z14 and later

This publication also includes the performance information for products that work closely with Db2 for z/OS, including IDAA and IBM Db2 for z/OS Data Gate, and new for Db2 13, Db2ZAI.

Most of the Db2 instrumentation data, including accounting, statistics, and performance traces, were formatted by using IBM Tivoli® OMEGAMON® for Db2 Performance Expert on z/OS 5.5.0.

The performance measurements in this book were generated primarily by members of the Db2 for z/OS Performance Team by using the z/OS system environment at the IBM Silicon Valley Laboratory. To illustrate the performance characteristics and usage considerations of Db2 12 and Db2 13, the performance measurements were conducted in a highly controlled environment. Although we strive to provide the best comparison between the baseline and new features, performance results depend on many variables, including the measurement environment, configuration setting, conditions, and workload characteristics that are used. As with any performance discussions, your results can vary.

### 1.1.1 Key Db2 13 features

This section describes the key features of Db2 13. For more information about new Db2 13 functions, see the Db2 13 product documentation and *IBM Db2 13 for z/OS and More*, SG24-8527.

#### **Business insights through SQL Data Insights**

SQL Data Insights is an optional feature of Db2 13 for z/OS. It provides built-in AI functions to perform semantic queries through the SQL language to enhance your ability to make business decisions.

The SQL Data Insights user interface and training services provide a quick way to trigger a simple training process, referred to as *Enable AI Query*, against Db2 tables and views. After training is complete, you can invoke three AI functions (**AI\_SIMILIARY**, **AI\_SEMANTIC\_CLUSTER**, and **AI\_ANALOGY**) from any SQL interface to perform semantic queries based on Natural Language Processing to discover the relationship between the relational rows and columns.

Both the training and query processes leverage IBM zSystems Integrated Information Processors (zIIPs) and invoke the IBM Z Deep Neural Network Library (zDNN) to leverage IBM zSystems hardware optimization to speed up the operations. Because the capability to embed the semantic query into SQL did not exist previously, the performance evaluation of this feature focuses on the scalability and factors that influence the performance of the Enable AI Query function and AI query execution.

#### **Scalability and performance enhancements**

Db2 13 continues to extend the performance and scalability improvements that started in Db2 12.

##### ***Index access improvement***

Fast index traversal was introduced in Db2 12, and it optimizes direct index access by storing non-leaf index information in a memory structure that is called *fast traversal blocks* (FTBs). Based on the name of the in-memory structure, this performance optimization feature is often referred to as the *FTB feature*.

Since the introduction of Db2 12, both through the continuous delivery stream and in Db2 13, fast index traversal has been enhanced to increase the candidate indexes that are eligible for FTB. We started by extending FTB support to unique indexes with a maximum key length of 64 bytes. Later, we improved how indexes with “include” columns are stored in FTBs, and added support for non-unique indexes through the `FTB_NON_UNIQUE_INDEX` subsystem parameter in V12R1M508 (function level (FL) 508). To control the usage of FTB features to specific indexes, a new option was added to define these selective indexes.

In Db2 13, non-unique indexes become FTB candidates by default. In Db2 13 FL 500, FTB support is further extended to indexes with up to a 128-byte key length for unique indexes and 120 bytes for non-unique indexes.

The optimization through FTBs is effective for workloads that access databases through indexes in a random key order. Therefore, the FTB feature tends to benefit Online Transaction Processing (OLTP) workloads with many simple lookups or random key inserts or updates.

For workloads that use sequential index access, an optimization that is called *index look-aside* can be used to improve performance. When Db2 determines that index access is performed sequentially, Db2 caches index leaf pages and uses the cached pages to avoid repeated access to the same pages. Db2 13 enhanced the triggering decision for the index look-aside feature for `INSERT`, `UPDATE`, and `DELETE` based on the real-time execution statistics instead of catalog statistics.

With these two optimizations in Db2 13, index access in both sequential and random order should see an excellent performance improvement.

### ***Concurrency improvement for partition-by-growth table spaces***

The space search algorithm for inserting rows into a partition-by-growth (PBG) table space is improved in Db2 13 when the insert involves multiple partitions. The algorithm in Db2 13 remembers the partitions that it previously failed to search, and the result is performance improvement and reduction of the unsuccessful inserts. The enhancement also uses real-time statistics (RTS) information to improve the cross-partition space search.

### ***Scalability at the system level***

To ensure continuous scalability with an ever-increasing volume of threads and Db2 objects, Db2 13 delivered enhancements in thread memory management. The allocation of the extended common service area (ECSA) by the Instrumentation Facility Component (IFC) has moved mostly to 64-bit storage to relieve common storage usage by Db2.

The scalability of partition-by-range (PBR) table spaces that use relative page numbering (RPN) is enhanced to reduce global contention when you use row-level locking in a data-sharing environment. With this enhancement, you benefit from more flexible and scalable partition management.

Db2 13 uses a z/OS 2.5 enhancement that moved a portion of data-set-related control blocks from 31-bit memory to 64-bit memory. As a result, you can keep more Db2 data sets open concurrently. Furthermore, you can leverage the new allocation option of z/OS scheduler work blocks (SWBs) in Db2 13 and further increase the number of data sets that you can open concurrently per Db2 subsystem.

### **Hybrid cloud experience**

Db2 13 improves the hybrid cloud experience with more granular application control, improved availability, a simplified migration process, and continuous security compliance.

### ***Application concurrency***

More granular lock control at the application level, instead of only at the system level, provides flexible concurrency control depending on the characteristics of the application. The application-level lock timeout and deadlock behavior can be controlled through a special register and global variable, and can be updated through application profiles without changes to the application.

In addition to application-level lock concurrency, a new profile **RELEASE** option was introduced to control the package release behavior for both local and remote applications. This enhancement increases the likelihood of Data Definition Language (DDL) break-in or maintenance operations, and reduces the impact to other running applications.

### ***Availability improvements***

Db2 12 for z/OS at FL 508 delivered the online conversion from a multi-table table space to PBG table spaces to reduce application outages and simplify the conversion process for DBAs.

To improve availability further, Db2 13 delivers online conversion from a PBG to a PBR table space. Db2 13 also eliminates the disruption when deleting the active logs from the bootstrap data set (BSDS). You can use the new **-SET LOG REMOVELOG** command to delete unneeded active log data sets without stopping Db2 subsystems.

The maximum size of Db2 directory table spaces SPT01 and SYSLGRNX was increased to 256 GB to allow more Db2 applications and data sets activities and reduce the need for frequent **REORG** statements after Db2 directory clean-ups.

### ***Simplified migration***

The process of migrating to Db2 13 is related to the Db2 12 continuous delivery process, which minimizes the disruptions from the upgrade process as much as possible. In Db2 13, the functional upgrade and catalog upgrade are controlled separately. You can migrate to Db2 13 without catalog updates and start taking advantage of the enhancements immediately.

### ***Continuous compliance***

In a hybrid cloud environment, providing evidence for security compliance becomes an important task for many Db2 customers. Db2 13 is enhanced to generate System Management Facility (SMF) type 1154 trace records for the recommended system security parameter settings. This information is used by the IBM Z Security and Compliance Center. Db2 13 with z/OS 2.5 caches successful execution of plan authorization checks when using the access control authorization exit (DSNX@XAC). In addition, the way information is stored in the plan authorization cache is enhanced to allow more entries to be stored for the same plan authorization cache size. These enhancements reduce the performance impact of plan authorization checking.

### ***Greater insights into managing a complex enterprise system***

Db2 13 delivers deeper operational insights in performance diagnostics and monitoring.

### ***Performance monitoring updates***

Db2 13 delivers several important updates to help with performance analysis.

The default statistics interval was changed from 60 to 10 seconds through the **STATIME\_MAIN** subsystem parameter to record more granular system-level performance information.

Db2 statistics trace information now includes information about the residency time for pages in the GBP in the CF with the IBM z16™, Coupling Facility Control Code Level (CFCC Level) 25 and later. You can use this information to adjust the size of GBP.

A new exception trace record, Instrumentation Facility Component Identifier (IFCID) 396, was added to the Db2 statistics to report on elongated index split operations. Combined with several new fields that are related to index split operations in RTS, you can easily identify the objects that are impacted by index split processing and possibly address the issue.

Distributed connections and thread monitoring are also enhanced. The Distributed Connection Control (DCC) feature of Db2ZAI 1.5 uses this new information to filter and control the distributed threads thresholds at the client application level or client user ID level.

Accounting trace (IFCID 3) provides new serviceability fields to record the longest lock or latch suspension and the resource for which the thread is waiting. This enhancement speeds up the analysis of slowdowns when the thread is holding or waiting for certain resources.

### ***Utility history***

Monitoring utility run times becomes easier and simpler by using the new Db2 13 utility history feature that collects essential information about Db2 utility run times in real time. Before Db2 13, monitoring and analyzing utility run times was challenging and cumbersome because the job logs from each run time were the main source of analysis. Utility processes write Db2 accounting trace information, but critical information such as the type of utilities that are running is missing. By using the new utility history feature, you can monitor utility run times for failures or delays and take appropriate corrective actions. The historical information can be used to tune or balance utility workloads.

## **1.2 High-level performance expectations**

The performance improvements for Db2 workloads running on Db2 13 after migrating from Db2 12 vary depending on the workload characteristics. The high-level performance expectation of Db2 13 is equivalent or better compared to Db2 12. The details of this evaluation are described in Chapter 4, “Workload-level measurements” on page 149.

At a high level, compared to the same workload running on Db2 12 with default subsystem parameters (other than the parameters that are related to the cache/pool), you might see the following improvements:

- ▶ OLTP: Equivalent, or up to a 5% CPU reduction by using the default subsystem parameters
- ▶ Relatively complex set of queries: Equivalent
- ▶ Update intensive batch processing: Equivalent, or up to a 10% CPU reduction
- ▶ Utility: Equivalent for most utilities, and up to a 60% CPU reduction for online **REORG** by using default subsystem parameters

## Online Transaction Processing workloads

The range of improvement in OLTP workloads varies depending on the setting of the FTB-related subsystem parameters: **INDEX\_MEMORY\_CONTROL** and **FTB\_NON\_UNIQUE\_INDEX**. Our measurements use the default values. In Db2 12 evaluations, the fast index traversal feature for unique indexes is enabled, but not for non-unique indexes. In Db2 13, more indexes become eligible for FTBs because both unique and non-unique indexes are enabled for FTB optimization by default, in addition to the expanded key length. Most of the improvements in OLTP workloads are attributed to the getpage reduction from indexes that are triggered by more FTB usage. Due to more FTB usage, the workloads observe an increase in 64-bit memory usage in the DBM1 address space.

## Querying workloads

Earlier Db2 releases introduced major improvements in access path optimization for complex queries. In Db2 13, we expect no significant access path changes when you migrate to Db2 13. A few optimizations are done in a Db2 Relational Data System (RDS) sort that do not influence the access path, as described in Chapter 8, “Query performance” on page 271. Unless you have sort-intensive queries, we expect equivalent performance between Db2 12 and Db2 13 with or without rebinding.

## Update-intensive workloads

We expect noticeable improvement for sequential batch insert, delete, or update processes from the index look-aside improvements. The batch jobs that trigger random access to the index access might benefit from the expanded fast index traversal feature.

If the batch involves inserts into PBG table spaces that cross partitions, you might see a reduction in getpage and corresponding CPU time reduction, as described in Chapter 7, “Application concurrency” on page 241.

## Db2 utilities

We observed improvement with **REORG INDEX** because Db2 automatically enables the NOSYSUT1 feature in Db2 13. As described in Chapter 9, “IBM Db2 for z/OS utilities” on page 279, the NOSYSUT1 enhancement can provide elapsed and CPU time reduction by avoiding the usage of the SYSUT1 data set for **REORG INDEX** operations. This feature was introduced in Db2 12, but is disabled by default.

If you are using inline statistics for **REORG** or **LOAD**, page-level sampling is supported and enabled based on the default value of the **STTPGSAMP** subsystem parameter in Db2 13. The default value of **SYSTEM** triggers page sampling that can result in performance improvement.

We expect equivalent performance for other utility types.

## 1.3 Synergy with the IBM Z platform

IBM Z technology continues to evolve based on the demand from your workloads and remains as an essential platform for your digital transformation. Db2 for z/OS continues to leverage the innovations from IBM zSystems hardware and software. This section describes the close synergy between Db2 and the IBM Z platform.

### 1.3.1 IBM zSystems hardware leverage

The following sections show how Db2 for z/OS has leveraged the enhancements that were delivered in the past several releases of IBM zSystems hardware.

## IBM z16

The IBM z16 brings AI and cyber resiliency to your hybrid cloud. In addition, the CF has several important improvements that affect Db2 performance in a data-sharing environment. The following list describes the key features from IBM z16 that Db2 leverages explicitly or implicitly:

- ▶ Improved single instruction, multiple data (SIMD) and on-chip AI accelerator  
SQL Data Insights in Db2 13 takes advantage of improved AI capability in IBM z16. SQL Data Insights training operations use the SIMD capability on IBM zSystems processors for vector processing. Db2 AI queries invoke the IBM Z AI Optimization library (zAIO) and IBM z/OS OpenBLAS libraries to leverage AI optimization between SIMD and on-chip accelerator based on the request.
- ▶ GBP residency time  
CFCC level 25, which was delivered in IBM z16, reports new cache residency times to its clients. These metrics show how long the data and directory entries remain in the cache structure. This information can be used as guidance to tune the GBP size. Db2 13 generates the information in the Db2 statistics trace records.
- ▶ CF scalability and coupling short link improvement  
IBM z16 CF images can scale higher than CF images on IBM z15. Extra performance improvements with CF short link (CS5) can produce CPU reduction for Db2 data-sharing configurations.
- ▶ SRB enhancements on z16 support the middleware restart boost in addition to initial program load (IPL) boost. Db2 can leverage the middleware restart boost.

## IBM z15

The IBM z15 delivers resiliency and performance. Db2 12 and Db2 13 for z/OS take advantages of the following features that are delivered in IBM z15:

- ▶ The IBM Integrated Accelerator for IBM Z Sort (Z Sort) feature is added in IBM z15 and later to speed up sort-intensive workloads. Both z/OS DFSORT and IBM Db2® Sort for z/OS leverage this feature, and the Db2 REORG utility can leverage it. Separately, Db2 RDS sort (**ORDER BY** and **GROUP BY**) also leverages Z Sort through **SORT LISTS (SORTL)** instructions starting with Db2 12 and later.
- ▶ DEFLATE-Conversion Facility (on chip compression) on IBM z15 replaces IBM zEnterprise® Data Compression (zEDC) Express. It is used by Db2 LOB compression with improvements in performance and scalability compared to zEDC Express.
- ▶ SRB at IPL offers extra capacity during system restart, including Db2 restart during the boost periods.
- ▶ Encrypted Diagnostic Data on IBM z15 provides encryption of diagnostic information. Db2 12 and later takes advantage of this feature to encrypt buffer pool contents in a Db2 dump.



## IBM z14

The IBM z14 was designed to be the most secure platform. Db2 12 and later leverage the improved encryption performance in z14 and z/OS encryption support to encrypt Db2 data that is provided by the following features:

- ▶ Performance improvements to Crypto Express and CP Assist for Cryptographic Functions (CPACF) and their capability to encrypt CF information made it possible to encrypt Db2 data at rest without significant performance impact.
- ▶ Three times more memory compared to IBM z13® means that Db2 can use more memory optimizations to reduce the operational cost. Db2 12 and later can leverage more memory through FTBs, contiguous buffer pools, and other in-memory optimizations.
- ▶ The Huffman compression algorithm delivers deeper compression of Db2 table spaces than the traditional fixed-length compression and can be leveraged by Db2 12 at FL 504 and later.
- ▶ IBM zHyperLink, a direct connect short distance between the processor and disk subsystem, offers low latency. Db2 12 and later leverages zHyperLink read/write capability for database synchronous reads and active log writes.
- ▶ Asynchronous cross-invalidation is delivered by CFCC level 23 and later. Db2 12 and later leverages the capability to optimize the cross-invalidation in data-sharing environments. This feature effectively reduces the overhead of cross-invalidation in a sysplex environment where long distance is involved.

### 1.3.2 Synergy with the IBM z/OS operating system

IBM z/OS provides a stable, secure operating system that is unique to the IBM Z platform. This section describes the specific z/OS features from which Db2 benefits:

- ▶ zDNN provides a set of APIs to enable deep learning operations that leverage IBM zSystems hardware optimization. SQL Data Insights in Db2 13 leverages zDNN, which is provided in z/OS 2.4 and later.
- ▶ Below-the-bar (BTB) constraint relief for data-set-related storage in z/OS 2.5 benefits both Db2 12 and later to increase the number of data sets that you can concurrently open per Db2 subsystem. Db2 13 further leverages the z/OS 2.5 **SWBSTORAGE (ATB)** option to maximize this benefit, and allows Db2 to raise its **DSMAX** value.
- ▶ The IBM RACF® authorization cache in z/OS 2.5 delivers performance improvements in access control authorization exit processing.
- ▶ IBM Z Security and Compliance Center automates evidence collection and validations to simplify compliance reporting. Db2 13 participates in this task by generating the compliance report that is related to the Db2 environment.

## 1.4 Db2 for z/OS and its ecosystem performance

In addition to IBM Z platform synergy, Db2 for z/OS works with an extensive set of products that together form an integrated ecosystem. These products include Db2ZAI, an extensive portfolio of development and administrative tools, IDAA, and IBM Db2 for z/OS Data Gate. This book describes the latest performance information for IDAA, IBM Db2 for z/OS Data Gate, and Db2ZAI in this book. A few notable features include IBM Integrated Synchronization technology and SQL optimization.

## 1.4.1 IBM Db2 Analytics Accelerator and IBM Integrated Synchronization

IBM Integrated Synchronization delivers a replication technology that is used by IDAA. Integrated Synchronization offers a performance advantage by reducing both the replication latency and CPU time during the log read operations. The same technology is used by IBM Db2 for z/OS Data Gate to replicate and synchronize Db2 for z/OS data to the target Db2 or Db2 Warehouse tables in the cloud in near real time. For more information about IDAA and IBM Db2 for z/OS Data Gate performance, see Chapter 10, “Performance enhancements for IDAA for z/OS and IBM Db2 for z/OS Data Gate” on page 307.

## 1.4.2 IBM Db2 AI for z/OS

Db2ZAI helps to improve your operational efficiency by learning from runtime information and infusing AI throughout. It supports the following three features:

- ▶ SQL optimization to discover and apply unique access path improvements
- ▶ System Assessment (SA) and Performance Insights to learn and simplify Db2 performance tuning by providing actionable recommendations
- ▶ DCC to monitor and control inbound network requests to the Db2 for z/OS server

### Access path improvement through SQL optimization

SQL optimization learns from your SQL run times and generates a more optimal access path that is based on AI predictions. The performance measurement of queries that use a set of host variables demonstrate noticeable performance improvements after Db2ZAI updates the access path. The strength of Db2ZAI is generating the new access path and self-healing as needed after the recommended access path is used to reduce the risk of new access paths impacting your workloads.

For more information about Db2ZAI and its capacity requirements, see Chapter 13, “IBM Db2 AI for z/OS benefits and capacity requirement” on page 365.

## 1.5 How to use this book

This publication describes the detailed performance measurements from Db2 13 for z/OS and Db2 12 for z/OS since general availability (GA). The book also includes the latest performance information on Db2 for z/OS ecosystems, including IDAA, IBM Db2 for z/OS Data Gate, Db2ZAI, and IBM zSystems hardware.

The performance measurements in this book are meant to show how the various enhancements in Db2 13 and its ecosystem perform. The topics that are covered are not limited only to performance enhancements. This book also describes usage considerations that were observed through the measurements, reasonable expectations for the enhancements, and a general understanding of the new functions.

All measurements were conducted in a laboratory environment, and as such they might be atypical because they were run in a dedicated and controlled environment with no other workloads that could potentially cause resource contentions, and in some cases focus on specific functions. Extrapolation and generalization require judgment.

The focus of the book is performance, and it does not cover all the features that are available in Db2 13. For a technical overview of Db2 13, see *IBM Db2 13 for z/OS and More*, SG24-8527.

This book includes the following chapters and appendixes:

- ▶ Chapter 1, “Introduction” on page 1  
This chapter presents the overview and general performance expectations in Db2 for z/OS 13.
- ▶ Chapter 2, “Scalability and availability” on page 13  
This chapter presents descriptions of scalability enhancements in Db2 13 and their impact on performance.
- ▶ Chapter 3, “Synergy with the IBM Z platform” on page 43  
This chapter describes performance evaluations of IBM zSystems hardware or software that became available since *IBM Db2 12 for z/OS Performance Topics*, SG24-8404 was published. It includes the measurements from Db2 12 and Db2 13, and covers IBM z14 to z16 evaluations.
- ▶ Chapter 4, “Workload-level measurements” on page 149  
This chapter describes performance evaluations and comparisons for a wide range of IBM and SAP benchmark workloads. We use Db2 12 as the baseline to evaluate Db2 13 performance.
- ▶ Chapter 5, “Data sharing” on page 187  
This chapter describes data-sharing-related enhancements in Db2 13. It also contains observations from the special study that was conducted to understand the performance impact of data-sharing configurations with a distance up to 49 kilometers.
- ▶ Chapter 6, “SQL Data Insights” on page 211  
This chapter describes the performance evaluation of the SQL Data Insights feature that was delivered in Db2 13.
- ▶ Chapter 7, “Application concurrency” on page 241  
This chapter describes the enhancements that are related to application concurrency in Db2 13.
- ▶ Chapter 8, “Query performance” on page 271  
This chapter describes Db2 13 enhancements that are related to query workloads.
- ▶ Chapter 9, “IBM Db2 for z/OS utilities” on page 279  
This chapter describes continued performance improvements in utilities in both Db2 12 and Db2 13.
- ▶ Chapter 10, “Performance enhancements for IDAA for z/OS and IBM Db2 for z/OS Data Gate” on page 307  
This chapter describes the major performance improvements in IDAA functions and IBM Db2 for z/OS Data Gate performance.
- ▶ Chapter 11, “Installation and migration” on page 331  
This chapter provides measurements of the Db2 migration process and considerations when migrating to Db2 13. The chapter also covers the performance-related subsystem parameter changes.
- ▶ Chapter 12, “Monitoring and instrumentation” on page 343  
This chapter includes IFCID updates that are related to performance features and monitoring guidelines.

- ▶ Chapter 13, “IBM Db2 AI for z/OS benefits and capacity requirement” on page 365  
This chapter describes how Db2ZAI can improve operational efficiency, and the Db2ZAI capacity requirement.
- ▶ Appendix A, “IBM Db2 workloads” on page 403  
This appendix describes the details of the workloads that are used to evaluate Db2 performance.
- ▶ Appendix B, “Artificial intelligence semantic queries” on page 409  
This appendix covers the AI semantic queries that run in the evaluation that is described in Chapter 6, “SQL Data Insights” on page 211.
- ▶ Appendix C, “IBM Db2 Analytics Accelerator for z/OS workloads” on page 421  
This appendix covers the workloads that run in the evaluation that is described in Chapter 10, “Performance enhancements for IDAA for z/OS and IBM Db2 for z/OS Data Gate” on page 307.



## Scalability and availability

Scalability and availability are major focus areas for new Db2 releases over the years. With both the daily growth of data volumes in Db2 and the increasing processing power of each new generation of IBM zSystems servers, constraints in Db2 can hold back workload scalability if they are not addressed. One of the more urgent constraints to address was the below-the-bar (BTB) private storage usage, which can throttle thread concurrency in a Db2 subsystem.

This chapter includes the following topics:

- ▶ Below-the-bar local agent storage reduction
- ▶ Extended common service area storage reduction
- ▶ Storage manager contraction optimization in Db2 13
- ▶ Raising the DSMAX limit
- ▶ Reduced security manager contention for Db2 access
- ▶ Fast index traversal enhancements
- ▶ DDF improvements

## 2.1 Below-the-bar local agent storage reduction

At the time of writing, Db2 subsystems can support 5,000 - 10,000 concurrent threads. As systems and workloads grow, the number of concurrently running threads, which often use dynamic SQL, increases, and the likelihood of hitting limitations for BTB private storage also increases. Even though much thread-related storage was moved above the 2 GB bar, Db2 12 still uses 31-bit private subpools (BTB storage) to store certain thread-related information.

As the number of concurrently running threads increase, the system can run low on BTB storage. Therefore, moving agent local BTB storage to an above-the-bar (ATB) storage area lifts these local agent storage constraints, allowing Db2 subsystems to handle a larger number of concurrent users so that you no longer need to worry about application or system outages due to a BTB storage shortage.

In Db2 13, the SQL statement text and attribute string for **PREPARE** and **EXECUTE IMMEDIATE** are stored in agent-local private ATB storage to help support more concurrent threads. The amount of storage that is allocated for the SQL statement text is based on its length. For any specific thread, multiple dynamic SQL statements can be running, depending on the nesting level. Although the maximum SQL statement length is 2 MB, much more storage can be allocated in ATB storage. This enhancement ensures that Db2 remains available, reliable, and resilient.

### 2.1.1 Performance measurement

Performance data was collected from several workloads by using a non-data-sharing configuration to evaluate the performance benefits and effectiveness of moving the SQL statement text and attribute string to ATB storage. The workloads have the following characteristics:

- ▶ Distributed dynamic SQL workload (WL1)
- ▶ Distributed native stored procedure with static SQL (WL2)
- ▶ Local dynamic SQL workload (TPCHDYN - WL3)
- ▶ Local mix of static and dynamic SQL workload (TPCH30 - WL4)
- ▶ Dynamic SQL with large SQL statement text (1.6 MB) (WL5)
- ▶ Native stored procedure with a large variable (2 MB) to hold the SQL statement text on PREPARE (WL6)

More detailed descriptions of these workloads can be found in Chapter 4, “Workload-level measurements” on page 149 and Appendix A, “IBM Db2 workloads” on page 403.

Besides the local and distributed dynamic and static workloads (WL1), two special workloads also were used. The first one, WL5, is a modified version of the distributed dynamic SQL workload (WL1) that uses a large SQL statement text. The second one, WL6, is a modified version of the native stored procedure workload (WL2) that uses a large variable (2 MB) to evaluate the benefit of this feature.

The main metrics that were used were the CPU time per commit, internal throughput rate (ITR), and BTB storage usage of the Db2 DBM1 and DIST address spaces.

Table 2-1 on page 15 presents the percentage difference between Db2 12 and Db2 13 for these six workloads.

Table 2-1 BTB agent local storage reduction in % difference between Db2 12 and 13 from six workloads

Metrics	WL1	WL2	WL3	WL4	WL5	WL6
Total CPU per commit	0.00	0.00	0.07	-0.55	1.68	-18.92
ITR	3.51	0.05	NA	NA	-0.68	8.83
Total DBM1 Storage BTB	-1.99	-1.56	-1.73	0.91	-96.62	-96.07
Total DIST Storage BTB	-2.33	0.00	0.00	0.00	-0.64	-20.86

The runs revealed the following general results:

- ▶ No significant overhead was observed on the “regular” (first four) workloads, and BTB storage usage was reduced.
- ▶ For the dynamic workload with long SQL statement text that used 2 MB host variables, the total DBM1 BTB storage decreased by 96%, from 568 MB down to 11 MB.
- ▶ For the native stored procedure with the large variable workload (WL6), DBM1 BTB storage decreased by 96%, and there was a 20% reduction in total DIST BTB storage.

## 2.1.2 Conclusion

By moving the SQL statement text and attribute string to ATB storage, DBM1 BTB storage usage on Db2 13 is reduced. The larger the SQL statement text, or the larger the variable in a stored procedure to store the SQL text, the more DBM1 storage reduction can be expected. With this feature, you can scale up the number of application threads and have more concurrent threads without having to worry about storage resources.

## 2.2 Extended common service area storage reduction

In addition to a storage manager daemon to contract BTB storage and extended common service area (ECSA) storage that is described in 2.3, “Storage manager contraction optimization in Db2 13” on page 16, Db2 13 also reduces ECSA storage usage by moving some storage pools from ECSA BTB storage to 64-bit HVCOMMON ATB storage. These enhancements include moving Instrumentation Facility Component (IFC) structures and DDF storage from ECSA to HVCOMMON.

### IFC structures moved to HVCOMMON

In Db2 12, the IFC structures use a mixture of pools, including ECSA storage. Because 31-bit ECSA storage is limited, this usage can become a bottleneck to scaling workloads when turning on various sets of Db2 traces. Although the usage of ECSA storage by IFC structures is typically not significant, moving them out of storage pools, including ECSA, to HVCOMMON in Db2 13 helps to reduce ECSA consumption for two storage pools that are used by IFC by 80%. The use of ECSA storage for Instrumentation Facility Interface (IFI) buffers is typically requested by monitoring programs.

### DDF storage moved to HVCOMMON

In Db2 13, ECSA storage for processing DDF threads is moved to HVCOMMON storage. DDF storage is used by the Data Communications Resource Manager (DCRM) Queue Manager, and some other storage pools are moved to ATB HVCOMMON storage.

## 2.2.1 Performance measurement

To evaluate these changes in Db2 storage usage, the distributed IBM Relational Warehouse Workload (IRWW) workload was used (for more information about IRWW, see “IBM Relational Warehouse Workload ” on page 404). A series of performance measurements were conducted with 500, 1000, 1500, and 2000 threads in a non-data-sharing environment. Db2 storage consumption is related to the number of concurrent active threads in the Db2 subsystem. Therefore, different numbers of concurrent active distributed threads were used as the main variable during these tests. All measurements were done by using the same IBM z15 logical partition (LPAR) with z/OS 2.5.

### Measurement environment

The workload was run by using the following system configuration:

- ▶ IBM z15 LPAR with three general processors, two IBM zSystems Integrated Information Processors (zIIPs), and 32 GB of memory.
- ▶ z/OS 2.5.
- ▶ Db2 non-data-sharing environment running the following levels of Db2:
  - Db2 12 function level (FL) 510
  - Db2 13 FL 501
- ▶ Db2 statistics trace class (\*) and monitor trace for Instrumentation Facility Component Identifier (IFCID) (124) are turned on.

### Measurement results

The workload shows a consistent increase in real storage and thread storage as the number of concurrent threads increases. In the tested workload, Db2 12 uses only a small amount of ECSA, so the reduction in Db2 13 was minimal.

## 2.2.2 Conclusion

By moving some IFC structures and DDF storage from ECSA to HVCOMMON and ATB, Db2 13 offers some reduction of BTB storage usage. The storage that is allocated for IFI buffers to monitor Db2 performance is moved outside of ECSA for the customers with multiple online monitoring products.

## 2.3 Storage manager contraction optimization in Db2 13

Db2 13 optimizes storage contractions for various BTB and ATB storage areas. Db2 Storage Manager relieves (contracts) BTB storage usage by releasing unused storage with **FREEMAIN** requests to free the virtual storage. Storage contraction for ATB storage occurs when the Db2 storage manager component issues **IARV64 REQUEST=DISCARDATA** requests to discard real storage frames that are related to 64-bit (ATB) storage that it no longer needs, and releases them back to z/OS.

In Db2 13, the storage manager component monitors the virtual storage usage by the Db2 subsystem in BTB private storage areas and in ECSA. When certain thresholds are exceeded, a storage management daemon starts contracting the corresponding storage pools until the amount of virtual storage that is allocated falls below the thresholds. Efficient BTB and ECSA storage contraction helps decrease the likelihood of Db2 subsystems hitting storage limitations and allows more concurrent threads to run in a Db2 subsystem.



For ATB storage management, Db2 13 changes its behavior when the storage manager issues **IARV64 REQUEST=DISCARDATA** requests to optimize Db2 subsystem performance. The rest of this section focuses on this change.

In Db2 12, the default setting of the **REALSTORAGE\_MANAGEMENT** subsystem parameter is **AUTO**. With this setting, Db2 issues **IARV64 REQUEST=DISCARDATA** requests to contract 64-bit thread storage in the following situations:

- ▶ At thread deallocation
- ▶ If the thread is reused, every 120 commits during normal operation
- ▶ After 30 commits, if the number of available real storage frames of the LPAR are low and the Db2 subsystem is running in “contraction mode”

However, excessive concurrent **DISCARDATA** requests can be triggered by many threads terminating in a short period. This situation can have a negative performance impact, such as causing an increase in the Db2 ssnmMSTR address space System Recovery Boost (SRB) CPU time, triggering heavy contention for certain z/OS Real Storage Manager (RSM) spin locks. In some extreme cases, the resulting z/OS Enhanced System Queue Area (ESQA) or ECSA storage shortage can cause more serious system-wide problems, including systems ending up in a wait state.

For an explanation of Db2 12 real storage management behavior in detail, and best practices for using the **REALSTORAGE\_MANAGEMENT** system parameter, see [Db2 Real Storage Management and Recommendations](#).

In Db2 13, the **REALSTORAGE\_MANAGEMENT** subsystem parameter is removed. Db2 13 does not issue the **IARV64 REQUEST=DISCARDATA** request during thread deallocation or after any specific number of commits for threads that are reused. Instead, Db2 keeps the 64-bit storage that is used by the terminated threads and reuses it when other threads need 64-bit storage. This change helps Db2 to avoid causing large bursts of concurrent **DISCARDATA** requests. To return unused ATB thread storage to z/OS in Db2 13, a timer-controlled storage contraction process issues **DISCARDATA** requests and releases unused real frames back to the operating system periodically.

With this new way of handling ATB thread storage in Db2 13, there should be fewer **DISCARDATA** requests because of the more efficient storage reuse. With the expectations of fewer **DISCARDATA** requests, the Db2 ssnmMSTR address space is expected to show some CPU savings. With proper storage reuse and centrally controlled storage contractions, Db2 13 should not consume much more ATB real storage than Db2 12 running with **REALSTORAGE\_MANAGEMENT=AUTO**.

### 2.3.1 Requirements

All the storage management changes take effect when you start running with Db2 13 code without dependencies on FLs.

### 2.3.2 Performance measurement

To evaluate the efficiency of the modified ATB storage contraction in Db2 13, a workload was created that is composed of complex static and dynamic queries, batch update jobs, and distributed transactions that run both static and dynamic SQL statements.

The two key performance indicators that were checked are the ssnMSTR address space SRB time and the total amount of storage that the Db2 subsystems used. The number of 64-bit storage **DISCARDATA** requests also were examined by using statistics from the IFCID 1 QSST section, as listed in Table 2-2.

Table 2-2 64-bit storage **DISCARDATA** statistics counters

Db2 version	How many pools were contracted? <sup>a</sup>	How many <b>DISCARDATA</b> requests were performed? <sup>a</sup>	How many virtual storage pages' frames were discarded through <b>DISCARDATA</b> requests?
Db2 12 <sup>b</sup>	QSST_P64DISNUM	QSST_P64DISBLK	QSST_P64DISPGS
Db2 13	QSST_P64PCTRCT	QSST_P64DISYES	QSST_P64DISPGS

a. Db2 12 and Db2 13 have different field names for the same counters.

b. For **REALSTORAGE\_MANAGEMENT=OFF**, the counters also are incremented and can be used as indicators of how many requests there would have been if **REALSTORAGE\_MANAGEMENT** was using a different setting than OFF. In reality, when running with **REALSTORAGE\_MANAGEMENT=OFF**, no **DISCARDATA** operations are performed.

## Measurement environment

The workload was run by using the following system configuration:

- ▶ IBM z14 LPAR with 16 general processors, four zIIPs, and 512 GB of memory
- ▶ z/OS 2.5
- ▶ Coupling Facility (CF): Two CFs at CF LEVEL 23, with three dedicated CPUs each
- ▶ Two-way Db2 data-sharing environment with two members running on the same LPAR running the following Db2 levels:
  - Db2 12 FL 510
  - Db2 13 FL 100

## Measurement scenario description

The workload was run with the following three configurations:

- ▶ The first configuration is with Db2 12 subsystems and the **REALSTORAGE\_MANAGEMENT** subsystem parameter set to OFF for both Db2 members.
- ▶ The second configuration is also with Db2 12, but the **REALSTORAGE\_MANAGEMENT** subsystem parameter is set to AUTO for both Db2 members.
- ▶ The third configuration uses Db2 13.

## Results

Db2 13, with its changed ATB storage management, achieves its goal of avoiding an excessive number of **IARV64 DISCARDDATA** requests compared to running with **REALSTORAGE\_MANAGEMENT=AUTO** under Db2 12:

- ▶ As shown in Figure 2-1, the number of **DISCARDDATA** requests from both Db2 members in the Db2 13 measurement dropped more than 99% compared to the Db2 12 **REALSTORAGE\_MANAGEMENT=AUTO** measurement. Matching with it, the number of pages that were discarded under Db2 13 was reduced by over 90%.

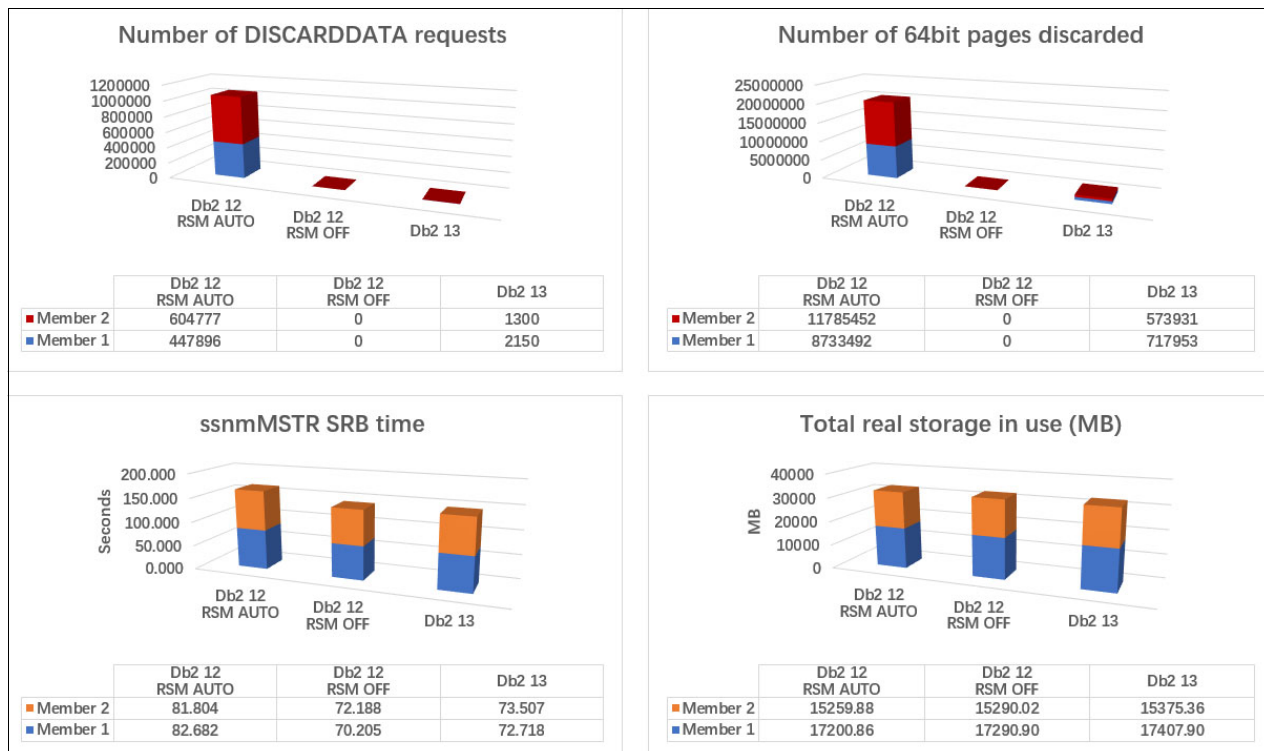


Figure 2-1 Storage manager above-the-bar contraction changes: key measurements metrics

- ▶ With the significant reduction in **DISCARDDATA** operations, the ssnmMSTR address space SRB CPU time was reduced 11 - 12% when comparing the Db2 13 measurement to the Db2 12 **REALSTORAGE\_MANAGEMENT=AUTO** measurement.
- ▶ Compared to the Db2 12 **REALSTORAGE\_MANAGEMENT=OFF** measurement, which does not perform **DISCARDDATA** at all, the Db2 13 measurement's ssnmMSTR SRB time increased by approximately 1 - 4%. Because the ssnmMSTR SRB time is typically a small portion of the total CPU time that is used by a Db2 workload, the overall impact of Db2 12 **REALSTORAGE\_MANAGEMENT=OFF** users should be minor.

The Db2 13 measurement also showed no increase in the real storage usage of the Db2 subsystem:

- ▶ All three measurements showed similar total real storage usage for the Db2 members. The differences are within the range of 0.04 - 0.3%. (The total real storage usage numbers are extracted from the IBM OMEGAMON for Db2 Performance Expert (OMPE) statistics reports' "REAL + AUX - DISC STORAGE IN USE – SUMMARY" section.)
- ▶ As shown in Figure 2-2, there are variations in real storage usage in different storage areas, although the total usage numbers are close. The variations are because of other storage-related changes in Db2 13, such as the BTB storage reduction and ECSA storage reduction that we described in 2.1, "Below-the-bar local agent storage reduction" on page 14 and 2.2, "Extended common service area storage reduction" on page 15.

In addition, the gradual database access thread (DBAT) contraction method in Db2 13, which is introduced in 2.7.2, "Gradual DBAT contraction" on page 40, can result in an increased high-water mark (HWM) for not-in-use disconnected DBATs, and might result in more storage being used overall until the DBATs that are no longer required are cleaned up.

**Note:** Because the LPAR real storage is not under stress, none of the three measurements trigger **KEEPREAL=NO DISCARDATA** operations.

Without IFCID 503 turned on, the DISCARDED STORAGE ELIGIBLE FOR STEAL is always recorded as 0. The total real storage in use numbers (REAL + AUX - DISC STORAGE IN USE) still includes the discarded pages and reflects the maximum amount of real storage that the Db2 subsystems used at any point during the measurements.

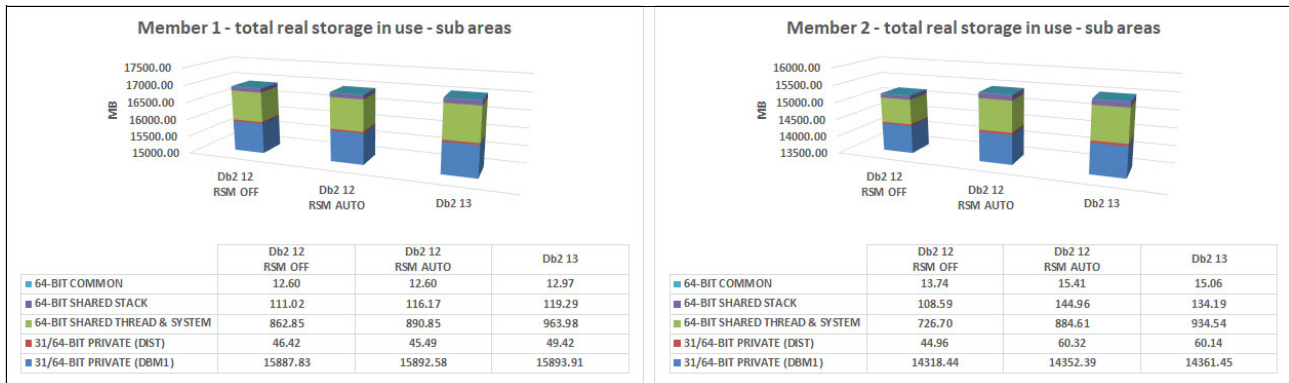


Figure 2-2 Storage areas that make up total real storage in use for the Db2 members

## Summary of results

The performance tests show that changes in Db2 13 to optimize the storage management of ATB thread storage by reducing unnecessary **DISCARDATA** processing are efficient, and overall real storage usage is similar to running Db2 12 with **REALSTORAGE\_MANAGEMENT=AUTO**.

## 2.4 Raising the DSMAX limit

Enterprise data growth, business consolidation, Db2 member consolidation, and conversion of Db2 segmented table spaces to partition-by-growth (PBG) table spaces increase the demand for more concurrent open data sets.

The **DSMAX** subsystem parameter specifies the maximum number of data sets that can be open concurrently. A **DSMAX** value of 200000 is the maximum that is allowed in both Db2 11 and Db2 12. This limit does not satisfy customer needs.

Two factors govern the limit of the number of concurrent open data sets in Db2:

- ▶ The required 31-bit DBM1 private memory per open data set
- ▶ The amount of 31-bit private memory remaining in DBM1 after excluding the required common memory (ECSA)

z/OS 2.5 provides an enhancement that moves a portion of the control blocks that are required for open data set to ATB storage. Most of these moves are transparent with Db2. In addition to these moves, in z/OS 2.5, dynamic allocation processing supports scheduler work blocks (SWBs) for data sets to use 64-bit storage, which reduces BTB storage usage. Moving SWBs to ATB storage requires functions that are provided in Db2 13. As a result, more concurrent data sets can be opened with z/OS 2.5 and Db2 13. Additionally, Db2 13 performance might be improved when opening many data sets concurrently by using z/OS 2.5.

### 2.4.1 Requirements

The following z/OS versions, Db2 version, and FLs are employed in this performance evaluation:

- ▶ IBM z14 LPAR with eight general processors, two zIIPs, and MT=1, with 4 TB of real storage
- ▶ z/OS 2.5 and z/OS 2.4
- ▶ Db2 13 at FL V13R1M100

## 2.4.2 Performance measurements

As shown in Figure 2-3, Db2 13 running on z/OS 2.5 has approximately 400 MB of BTB storage that is available in DBM1 when the number of concurrently opened data set reaches 400,000. With Db2 13 running on z/OS 2.4, DBM1 BTB storage is exhausted around 370,000 opened data sets. This comparison demonstrates the lean BTB storage advantage per open data set that z/OS 2.5 provides.

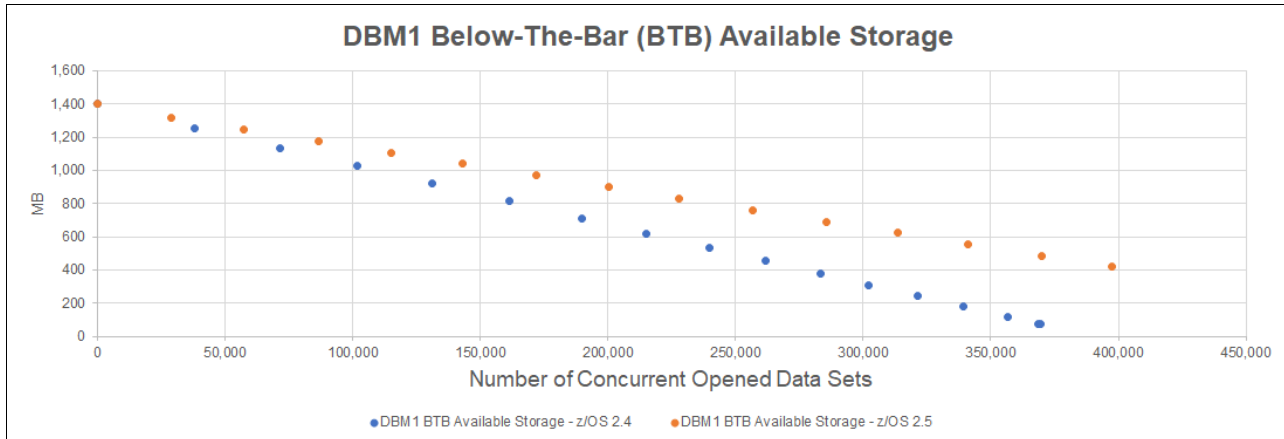


Figure 2-3 Db2 13 function level 100 DBM1 BTB available storage as opened sets that use z/OS 2.4 and z/OS 2.5

When Db2 DBM1 BTB storage is nearly depleted and only approximately 5% of the extended region size remains, Db2 issues a DSNV508I warning message (DSNV508I DSNVMON - DB2 DBM1 BELOW THE BAR STORAGE alert-level) to protect Db2 from failing. Any subsequent request to open a data set receives an intentional Db2 abend with a reason code of 00E20003 or 00E20016.

DBM1 BTB storage usage is equal to the sum of the following fields in the DBM1 and MVS Storage Below 2 GB section of the OMPE Db2 Statistics trace report:

- ▶ 31-Bit Extended Low Private
- ▶ 31-Bit Extended High Private

Figure 2-4 shows z/OS 2.5 allows more opened data sets than z/OS 2.4 with the same Db2 version opening data sets.

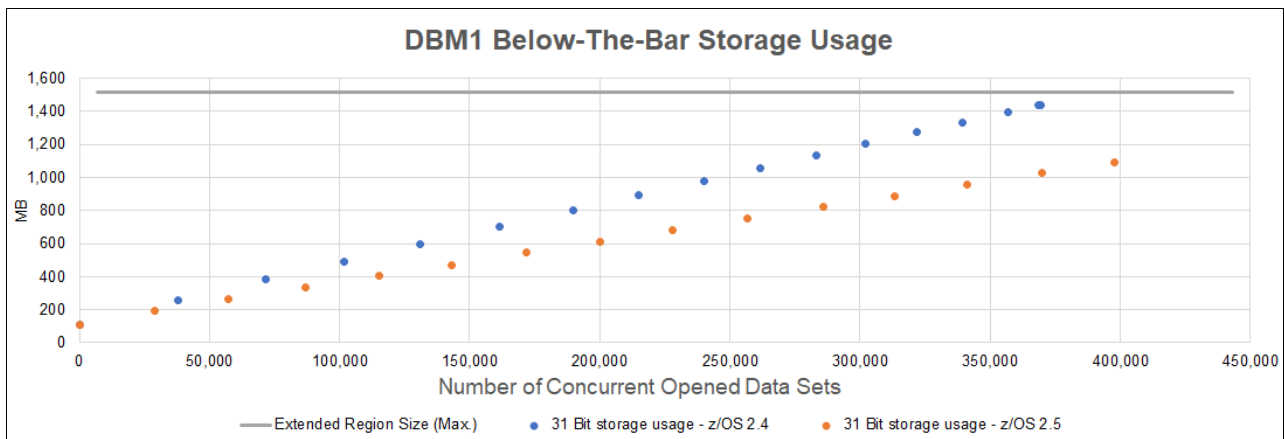


Figure 2-4 Db2 13 DBM1 BTB storage usage as opened data sets while using z/OS 2.4 and z/OS 2.5

The measurements that are shown in Figure 2-3 on page 22 and Figure 2-4 on page 22 illustrate that the minimum BTB storage requirement for an open data set depends on the z/OS version:

- ▶ Using z/OS 2.4, approximately 4 KB per data set
- ▶ Using z/OS 2.5, approximately 3 KB per data set

Figure 2-5 shows the Db2 13 FL 100 DBM1 64-bit storage in use, excluding the storage that is used by the Virtual Buffer Pool, for z/OS 2.4 and z/OS 2.5.

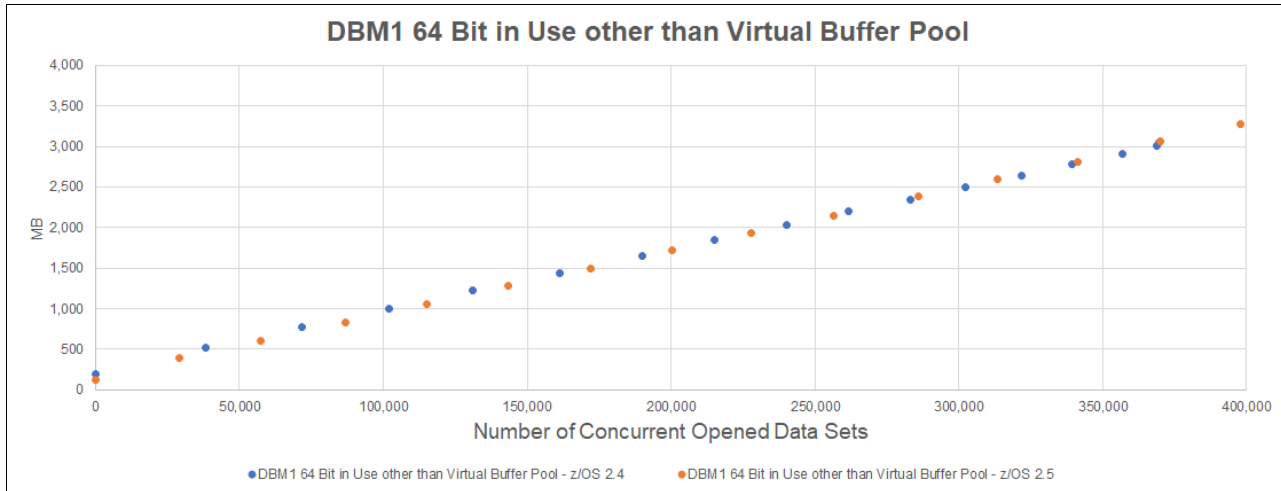


Figure 2-5 Db2 13 DBM1 64-bit storage usage excluding the virtual buffer pool

There is no appreciable difference between the two z/OS releases, except that the size stops growing around 370,000 open data sets for z/OS 2.4. The reason is that the DBM1 BTB storage is exhausted around this point for z/OS 2.4. For z/OS 2.5, ATB storage for the DBM1 address space continues to grow linearly until the limit of 400,000 open data sets is reached. We excluded the memory that is used by virtual buffer pools in Figure 2-5 to demonstrate a clear view of 64-bit storage usage.

The CPU cost to open a data set on the Db2 side is reflected in the DBM1 TCB CPU time. Figure 2-6 shows the cumulative DBM1 TCB time while Db2 13 FL 100 is opening data sets running on z/OS 2.4 and z/OS 2.5 respectively. Obviously, z/OS 2.5 is more efficient at opening data sets than z/OS 2.4.

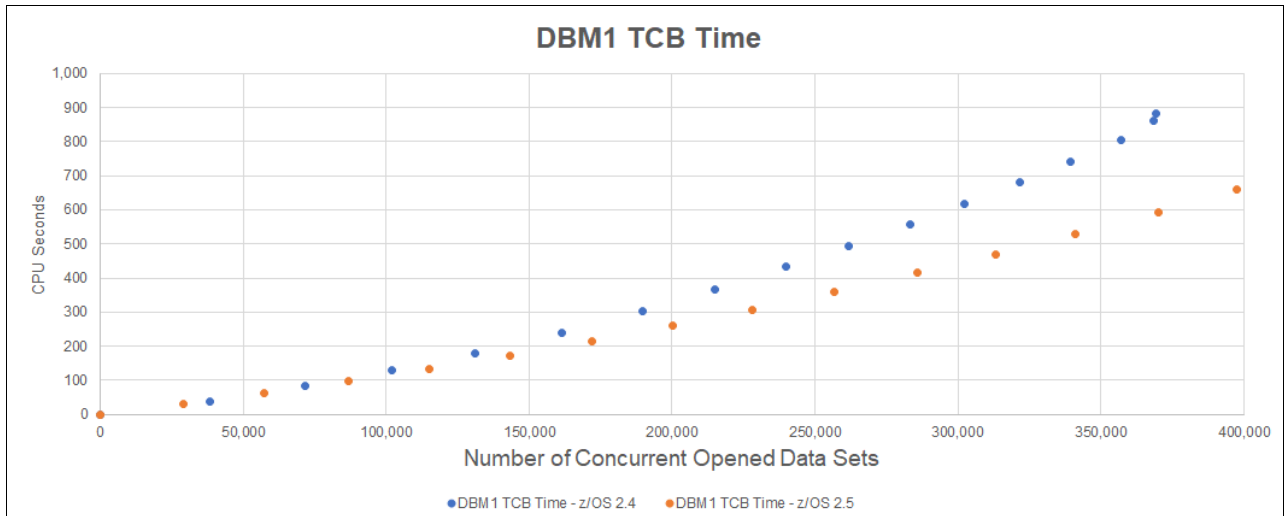


Figure 2-6 CPU cost of opening data sets on the Db2 side (reflected in DBM1 TCB time)

The elapsed time of Db2 13 FL 100 opening data sets by using z/OS 2.4 and z/OS 2.5 is shown in Figure 2-7. Again, there is no appreciable difference, except that z/OS 2.5 reaches the maximum of 400,000 data sets, and z/OS 2.4 falls short of the limit due to exhaustion of BTB DBM1 storage.

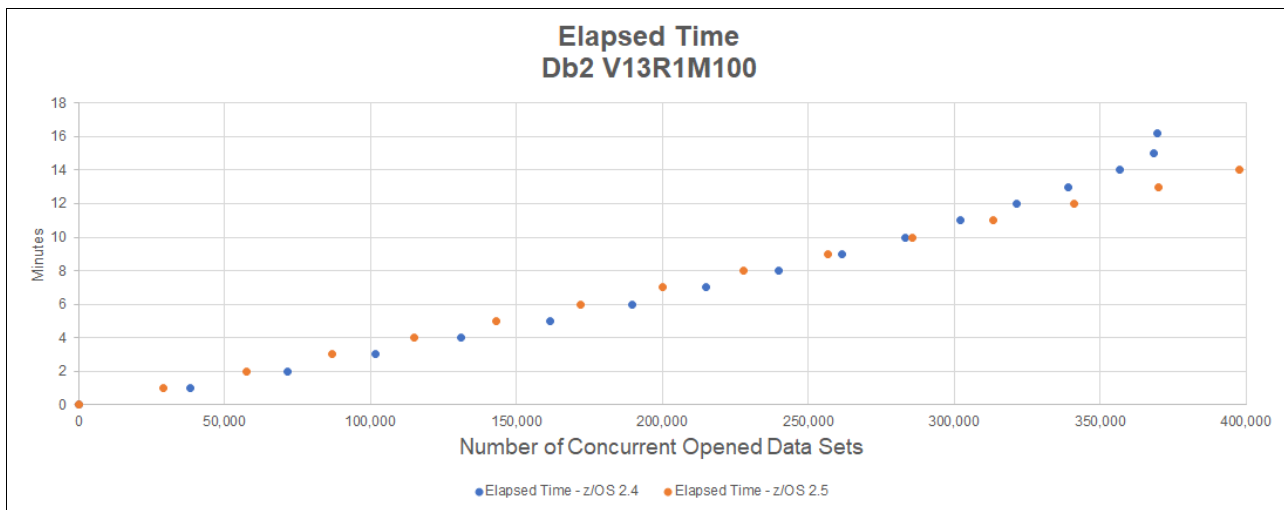


Figure 2-7 Elapsed time of opening data sets in z/OS 2.4 and z/OS 2.5



Figure 2-8 examines the opening of data sets from an LPAR point of view, which is broader than Db2. The following observations are noted:

- ▶ The CATALOG address space consumes nearly as much CPU time as the Db2 DBM1 address space.
- ▶ The GRS address space real storage usage grows significantly, as much as 2 GB or more in our measurements, during the measurement.
- ▶ Access to SYSIGGCAS\_ECS or ISGLOCK CF structures is approximately 10 requests per data set.

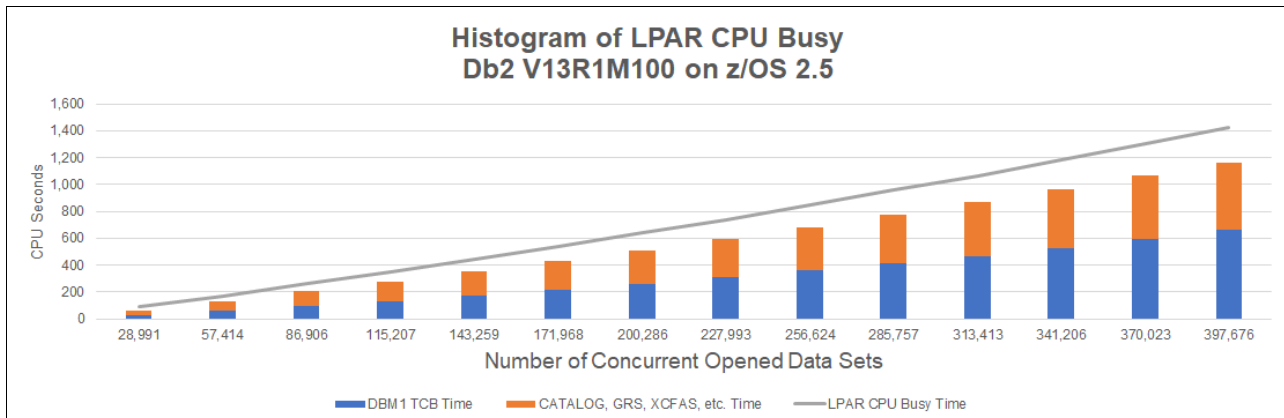


Figure 2-8 Histogram of LPAR CPU busy: Db2 13 function level 100 on z/OS 2.5

### 2.4.3 Monitoring information

Db2 accounting and statistics trace data, and RMF data, was used to monitor performance during measurement.

### 2.4.4 Usage considerations

To leverage this feature, use z/OS 2.5, and complete one of the following actions:

- ▶ Update the ALLOCxx parmlib member to SYSTEM SWBSTORAGE(ATB). This method remains in effect after the next initial program load (IPL).
- ▶ Issue the `'SETALLOCSYSTEM,SWBSTORAGE=ATB'` system command. This method is effective only until the next IPL.

The following new DSNB280I message is issued during Db2 13 start or restart if the new dynamic allocation function that supports SWB blocks for data sets in 64-bit storage is successfully enabled:

```
DSNB280I DYNAMIC ALLOCATION SUPPORT FOR 64-BIT RESIDENT SWB BLOCKS IS ENABLED
```

### 2.4.5 Conclusion

To summarize, with Db2 13, 400,000 data sets can be opened concurrently by using z/OS 2.5, and approximately 370,000 data sets can be open on z/OS 2.4 before Db2 fails.

**Note:** 400,000 concurrently opened data sets were achieved in a controlled environment at the Silicon Valley Laboratory. During this test, there were no other Db2 or system activities than opening the data sets. The DBM1 address space is clean and pristine, unlike most customer systems, with no other activity running in the system. Therefore, your results vary depending on your individual case.

SWBs are placed in 64-bit storage in z/OS 2.5, which meets the DBM1 BTB storage requirement.

The estimated BTB storage requirement for an open data set is approximately 3 KB bytes for z/OS 2.5, and 4 KB is used per open data set for z/OS 2.4.

Nontrivial collateral cost such as CATALOG address space CPU time consumption, GRS address space storage growth, and sysplex SYSIGGCAS\_ECS and ISGLOCK structures activity should not be overlooked.

Besides the enhancement in z/OS 2.5 to move the SWBs ATB, there are other enhancements that reduce the amount of storage that is needed to represent an open data set. As a result, Db2 12 running on z/OS 2.5 also can open more data sets than with z/OS 2.4. However, because Db2 12 cannot leverage SWB in ATB storage, expect less concurrently opened data sets than when using Db2 13.

## 2.5 Reduced security manager contention for Db2 access

Db2 13 includes many improvements that reduce contention for security manager resources when accessing Db2. One of those improvements is to leverage the Db2 external security caching support when using the access control authorization exit (DSNX@XAC).

### 2.5.1 Enabling plan authorization cache

This section describes enhancements to the plan authorization cache that is provided by Db2 13.

#### Hardware and software requirements

The plan authorization cache has the following software requirements:

- ▶ z/OS 2.5.
- ▶ Db2 13.
- ▶ The `AUTHEXIT_CACHEREFRESH` subsystem parameter is set to ALL.
- ▶ The access control authorization exit (DSNX@XAC) is active.

#### Plan authorization cache for external security

When you use the access control authorization exit (DSNX@XAC) for authorization checking, Db2 13 can cache the successful plan execution checks in the plan authorization cache. The entry with the primary authorization ID is cached when the access is allowed based on the profile in the RACF resource class for the plan (for example, MDSNPN or GDSNPN). The results are not cached if access is allowed due to administrative authority, such as DATAACCESS or SYSADM.

The plan authorization cache is enabled when the existing `AUTHEXIT_CACHEREFRESH` subsystem parameter is set to ALL.

Before Db2 13, when the **AUTHEXIT\_CACHEREFRESH** subsystem parameter is set to ALL and the access control authorization exit is active, Db2 listens to the type 62, type 71, and type 79 ENF signals from RACF for user profile or resource access changes and refreshes the Db2 cache entries as needed. Db2 13 also is enhanced to refresh the plan authorization cache entries for a particular plan when the user profile or resource access is changed in RACF and **SETROPTS RACLIST REFRESH** is issued for the RACF resource class for the plan, such as MDSNPN.

### ***AUTHCACH subsystem parameter***

The plan authorization cache size is set to 4 K by default, and the **AUTHCACH** subsystem parameter was removed to simplify cache management.

You can still specify the **CACHESIZE** bind option on the **BIND PLAN** subcommand to control the plan authorization cache size at the plan level (and override the default plan authorization cache size of 4 K).

### ***Statistics record counters***

The existing Db2 statistics trace counter **QTAUCCH** is incremented after a successful plan authorization check for both Db2 native and external security controls when the cache is used.

The following counters were added in Db2 13. These new counters are set regardless of whether you use Db2 native or external security controls.

- ▶ **QTAUCNOT** tracks the number of times that the plan **EXECUTE** privilege was checked and not found in the plan authorization cache.
- ▶ **QTAUCOW1** tracks the number of times that Db2 had to swap out an authorization ID in the plan authorization cache due to lack of space.

### ***Smarter algorithm for authorization IDs***

The algorithm that is used to calculate the number of authorization IDs per plan cache entry is enhanced to potentially allow for storing more IDs in the cache. This enhancement reduces the number of IDs that are swapped in and out of the plan authorization cache.

## **2.5.2 Performance measurement**

A Db2 13 performance improvement of 2% elapsed time and 8% CPU time was observed with a workload that used four authorization IDs accessing the same plan in a loop 500 times. This measurement was compared to the same authorization IDs accessing the same plan in Db2 12, where no plan authorization caching is available.

The improvement that is seen can vary depending on the number of authorization IDs that are involved and whether the authorization IDs are found in the cache.

## **2.6 Fast index traversal enhancements**

Fast index traversal (fast traversal blocks (FTB)) was first introduced in Db2 12. At Db2 12 general availability (GA), only unique indexes with key sizes smaller than or equal to 64 bytes were supported. Then, columns in the **INCLUDE** list were considered as part of the index key size limit of 64 bytes. You can learn more about how FTB helps improve performance, find more information about the feature's methodology, and see what the initial performance evaluations looked like at Db2 12 GA in Chapter 2, "Scalability enhancements and subsystem performance", in *IBM Db2 12 for z/OS Performance Topics*, SG24-8404.

Because FTB brings performance benefits for a wide range of workloads that access data randomly, it is natural for the Db2 to expand its usage to more indexes.

At Db2 12 FL 508, the following two improvements were introduced to expand FTB support to cover more indexes:

- ▶ The columns in the INCLUDE list of unique indexes are no longer built into the FTB structures, and their lengths do not count toward the size limit for the index key size, which enables a more unique index to be eligible for FTB usage.
- ▶ FTB functions also were expanded to support non-unique indexes. A non-unique index is deemed an FTB candidate when its key size for the columns is 56 bytes or less. The Db2 system task that monitors unique index FTB eligibility now also monitors candidate non-unique indexes to automatically allocate and deallocate FTB structures for these indexes. A new system parameter **FTB\_NON\_UNIQUE\_INDEX** was introduced to control whether non-unique index FTB usage is enabled for the Db2 subsystem. In Db2 12, the default value for **FTB\_NON\_UNIQUE\_INDEX** is NO, which means non-unique indexes do not have FTBs that are created for them.

In Db2 13 at FL 500 or later, two more changes are introduced to further expand FTB eligibility:

- ▶ The key size limit is raised from 64 bytes to 128 bytes for unique indexes, and from 56 bytes to 120 bytes for non-unique indexes.
- ▶ The default value of the **FTB\_NON\_UNIQUE\_INDEX** subsystem parameter is changed from NO in Db2 12 to YES in Db2 13, which means that FTB support for non-unique indexes is enabled by default in Db2 13 FL 500 or later.

Usability and monitoring enhancements for the FTB function also were introduced over the years. For more information, see 2.6.2, “Performance measurements for an FTB key size increase” on page 32.

## 2.6.1 Performance measurements for FTB support for non-unique indexes

To evaluate the benefits of expanding FTB support to non-unique indexes, we designed a series of random insert, update, and select performance workloads that access data through non-unique indexes.

### Measurement environment

The following environment was used for the FTB non-unique index support feature’s performance evaluations under non-data sharing:

- ▶ One IBM z14 LPAR with two general processors, one zIIP, and 64 GB of real storage
- ▶ A non-data-sharing Db2 12 subsystem running at FL 504
- ▶ z/OS 2.3
- ▶ A DS8870 direct access storage device controller

The following test environments were for the two-way data-sharing measurements:

- ▶ Two IBM z14 LPARs with two general processors, one zIIP for each LPAR, and 64 GB of real storage for each LPAR
- ▶ Two-way data-sharing Db2 12 subsystems running at FL 504
- ▶ z/OS 2.3

- ▶ DS8870 direct access storage device controller
- ▶ Two IBM z14 Internal Coupling Facility (ICF) LPARs running CFLEVEL 23 CFCC, which each LPAR connected to the z/OS LPARs through four Internal Coupling (ICP) channels

**Note:** The performance measurements were conducted at FL 504 with internal pre-release development libraries, which contained all the non-unique index FTB support code changes. However, the code was not released until FL 508, which is the lowest FL that supports a non-unique index to use FTB structures.

## Measurement scenario description

All the workloads access data randomly through a non-unique index. The base table space was partition-by-range (PBR) table space. The index keys were made up of fixed-length columns total of 56 bytes. We tested indexes with different numbers of rows for the same key values to make sure that FTB has performance benefits with different degrees of key duplication.

All of random select, update, and insert workloads were run under a non-data-sharing configuration. The random insert workload was run in a two-way data-sharing configuration.

The random select workloads perform 200 random selects per commit. There are three scenarios with a different number of data rows for each index key value:

- ▶ One row for an index key
- ▶ Ten rows for an index key
- ▶ One hundred rows for an index key

The random update workloads have two scenarios:

- ▶ One row for an index key. The workload updates one row per commit.
- ▶ Ten rows for an index key. The workload updates 10 rows per commit.

The random insert workloads have three scenarios with a different number of data rows for a single index key value:

- ▶ One row for an index key. The test inserts one row per commit.
- ▶ Ten rows for an index key. The test inserts 100 rows per commit.
- ▶ One hundred rows for an index key. The test inserts 100 rows per commit.

## Results

CPU improvements were observed for all the performance evaluations when FTBs were allocated for the non-unique index. This section describes the results in two parts: the results for the non-data-sharing environment are presented, and then followed by the results for the two-way data-sharing environment.

For the non-data-sharing tests comparing when FTB is used for the non-unique index to when FTB is not used, the following observations were noted:

- ▶ Getpage reductions of up to 72% for the test workloads.
- ▶ With the getpage reductions, there are Db2 total CPU usage savings of up to 17%. The total Db2 CPU time that is used includes the class 2 CPU time and Db2 address space CPU time that is spent on general central processors (CPs). The CPU time that is spent on zIIP is not included in the analysis. These FTB measurements do not affect zIIP usage for the workloads much. All the zIIP time is from Db2 address spaces and had only small changes.

- ▶ For the random select workload, there was an up to a 59.4% getpage reduction. However, the more rows that a single index key pointed to, the less getpage reduction in percentage was seen.
  - When there is a single row of data for each index key value without FTB, for each random select, there are four getpages on the 4-level index and one getpage on the table space to retrieve the data. With FTB enabled for the index, there is one getpage on the index and one getpage on the table space for a single select, which means that with FTB enabled, the theoretic getpage reduction should be 60%. Our test result of a 59.4% getpage reduction as shown as the first bar in Figure 2-9 verifies the theory.

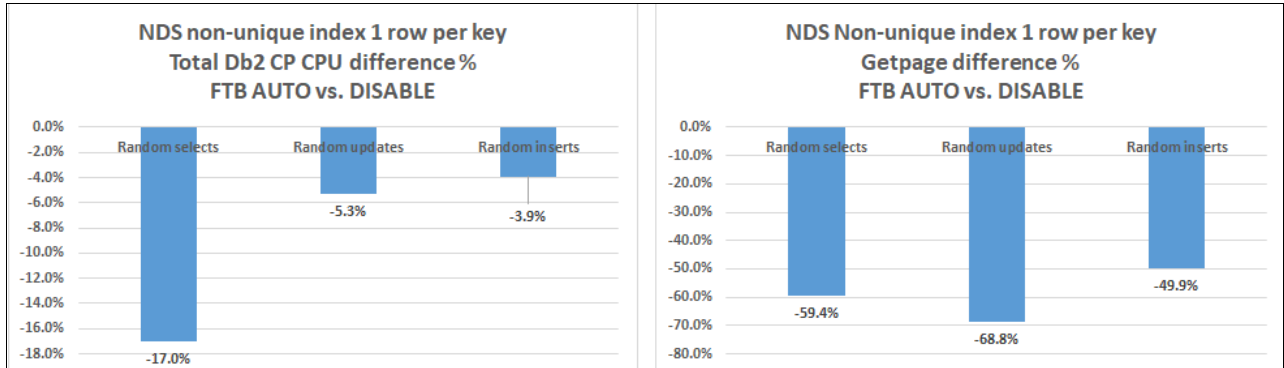


Figure 2-9 Non-unique index 1 row per key CPU and getpage changes: non-data sharing

- With 10 rows of data for every index key value without FTB, in our test there are four getpages on the 4-level index to locate the index key and six getpages on the table space to retrieve all 10 rows of data for each select. With FTB enabled for the index, for each select, there is one getpage on the index and six getpages on the table space. Therefore, in theory, the total getpage reduction FTB brings for a single select is 30%. Random selects getpage reduction of 29.4% in Figure 2-10 verifies the theory.

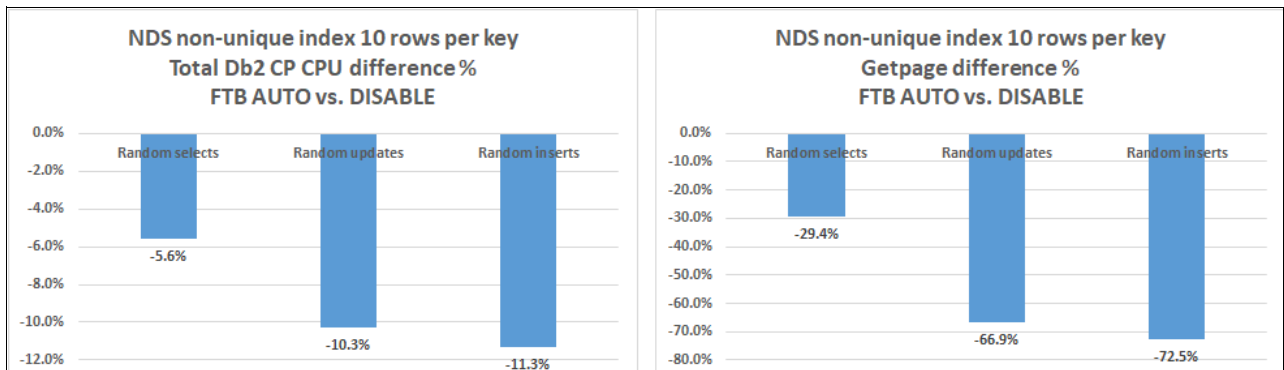


Figure 2-10 Non-unique index 10 rows per key CPU and getpage changes: non-data sharing

- When there are 100 rows of data for every index key value without FTB, in our test there are 4.2 getpages on the 4-level index (because there are 100 RIDs for a single index key, and one index key's data can span two pages) and 100 getpages on the table space for each select. With FTB enabled for the index, we see the index getpages reduced to 1.5 for a single select. As Figure 2-11 on page 31 shows, the total getpage reduction is 2.7%.

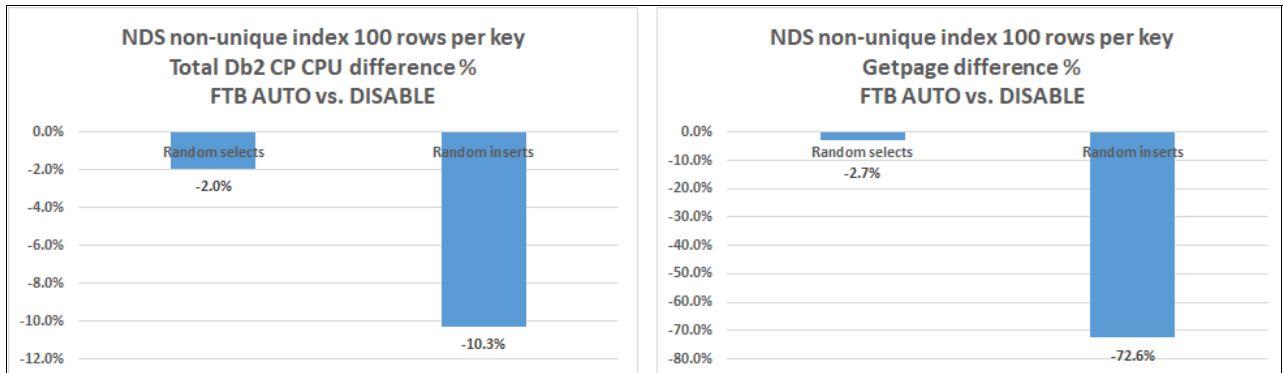


Figure 2-11 Non-unique index 100 rows per key CPU and getpage changes: non-data-sharing

- ▶ FTB data structures are allocated in a 64-bit variable storage pool. The biggest increase that we see in 64-bit variable storage pool usage is 294 MB for an FTB structure with a size of 119,921 KB, as the output of the `-DISPLAY STATS(IMU)` command shows.

For the two-way data-sharing insert tests (see Figure 2-12), comparing when FTB is used for the non-unique index to when FTB is not used.

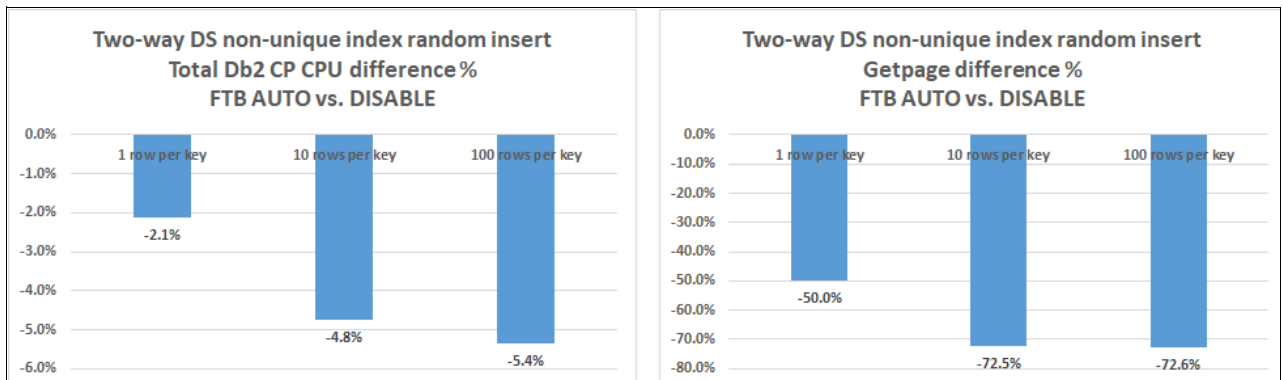


Figure 2-12 Non-unique index insert workload CPU and getpage changes: two-way data-sharing

- ▶ In the two-way data-sharing configuration, the insert workload showed similar degrees of getpage reduction as the non-data-sharing tests. The getpage reduction is as high as 72%.
- ▶ Db2 total CPU saving is observed for all the measurements with 1, 10, or 100 rows per index key value. The Db2 CPU reduction is in the range of 2.1% - 6.9%.
- ▶ The biggest 64-bit variable storage usage increase for a single Db2 member was 300 MB for the FTB with 95936 KB, as the output of the `-DISPLAY STATS(IMU)` command shows.

Notify messages between the two Db2 members increase up to 14 times compared to the test runs without FTB enabled. This increase occurs because index splits happen occasionally as data is inserted. With FTB enabled for the index, the Db2 member handling the index split sends notify messages to the other members to let them know about the FTB structure changes. This situation causes the random insert workload's class 3 suspension time for notify messages to increase. We also observed a slight increase of `ssnmIRLM` address space SRB time. The increase is far less than the class 2 CPU reduction.

Figure 2-13 depicts the impact of notify messages on the workload performance, taking the 100 rows per index key insert tests' data as an example.

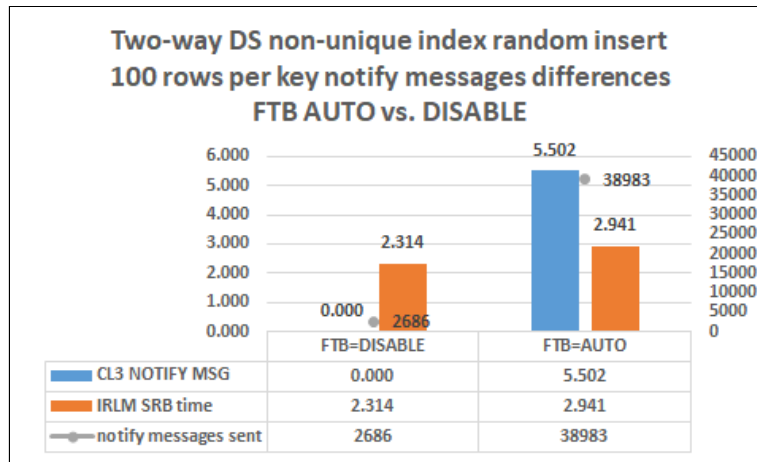


Figure 2-13 Non-unique index insert workload notify message differences: two-way data-sharing

### Summary of results

Allowing non-unique indexes to be accessed through FTB structures can help reduce getpage operations that are performed on the non-leaf index pages and save CPU time for workloads that access indexes randomly.

Although the performance evaluations that are described here cover only limited non-unique index usage scenarios, they are not the only tests that were run to make sure that FTB support for non-unique indexes was properly vetted. The Db2 for z/OS performance team also measured the feature's performance with the more regular workloads, such as the DPRB, TPC-E, relational transaction workloads (RTWs), query workloads, and others when it was first introduced. We made sure that no regression was seen for any of these workloads. In Chapter 4, "Workload-level measurements" on page 149, you can see various workloads showing performance improvements after migrating to Db2 13 with the `FTB_NON_UNIQUE_INDEX` subsystem parameter set to the default value of YES, which turns on FTB support for non-unique indexes.

## 2.6.2 Performance measurements for an FTB key size increase

Db2 13 enhanced the FTB support and extended the key length limit for eligible indexes as follows:

- ▶ For unique indexes, the key length can be up to 128 bytes.
- ▶ For non-unique indexes, the key length can be up to 120 bytes.

The Db2 for z/OS performance team conducted a series of tests to make sure that this FTB enhancement meets its functional and performance target and does not cause performance regression.

### Measurement environment

The performance measurement was conducted by using the following components:

- ▶ One IBM z14 LPAR
- ▶ z/OS 2.4
- ▶ Six general processors and three zIIPs



- ▶ 56 GB of storage
- ▶ Non-data sharing and two-way data sharing

### **Measurement scenario description**

The performance evaluations for this FTB enhancement include the following coverage:

- ▶ The performance impact of this enhancement on the existing FTB support with index key size of 64 bytes or less. The measurements were conducted as follows:
  - Two-way data sharing
  - Index key size: 24 bytes and 48 bytes
  - Table space types: PBG, PBR with absolute page numbering (APN), PBR with relative page numbering (RPN), and classic partitioned
  - Concurrency: Ten threads
  - Sequential insert, update, delete, fetch, random select, and insert
  - Db2 13 with and without this enhancement
- ▶ The performance impact of this enhancement on non-data-sharing cases with an index key size of 64 - 128 bytes. The measurements were conducted as follows:
  - Non-data sharing
  - Index key size: 96 bytes
  - Unique index or non-unique index
  - Focus on random select with index data access
  - Concurrency: Ten threads
  - Table space types: PBG, PBR, and RPN
  - Comparison point: Db2 13 with and without this enhancement
- ▶ The performance impact of this enhancement on data-sharing cases with an index key size 64 bytes - 128 bytes. The measurements are conducted as follows:
  - Two-way data sharing
  - Index key size: 96 bytes
  - Unique index or non-unique index
  - Concurrency: Ten threads
  - Sequential insert, update, delete, random select, and insert
  - Table space type: PBG
  - Comparison point: Db2 13 with and without this enhancement

## Results

CPU improvements were observed for all the performance evaluations when fast index traversal was used for the non-unique index. This section describes the results in two parts: For the tests in a non-data-sharing environment, and then for the two-way data-sharing tests.

For the non-data-sharing tests, when FTB was used for the non-unique index was compared to when FTB is not used:

- ▶ The performance results for test cases with an FTB index key size of 24 bytes and 48 bytes showed that this FTB enhancement introduced no noticeable performance degradation for the existing support and all the differences are in the noise range.
- ▶ Non-data sharing for an index key size of greater than 64 bytes: For an index with a key size greater than 64 bytes, such as 96 bytes, the index is not eligible for FTB with the existing FTB support, but it is eligible for FTB with this FTB enhancement. In this measurement, the total number of rows that are inserted is 20 million and the number of index levels is 4. The performance results for random select with index-data access are shown in Figure 2-14. This FTB enhancement improves CPU time by approximately 8% for a case with the unique index of 96 bytes, and 4 - 6% CPU time improvement is seen for a case with the non-unique index of 96 bytes. The CPU savings occur because with this enhancement, an FTB structure is created for the index, and for one random index access, only one getpage operation is needed instead of four without FTB.

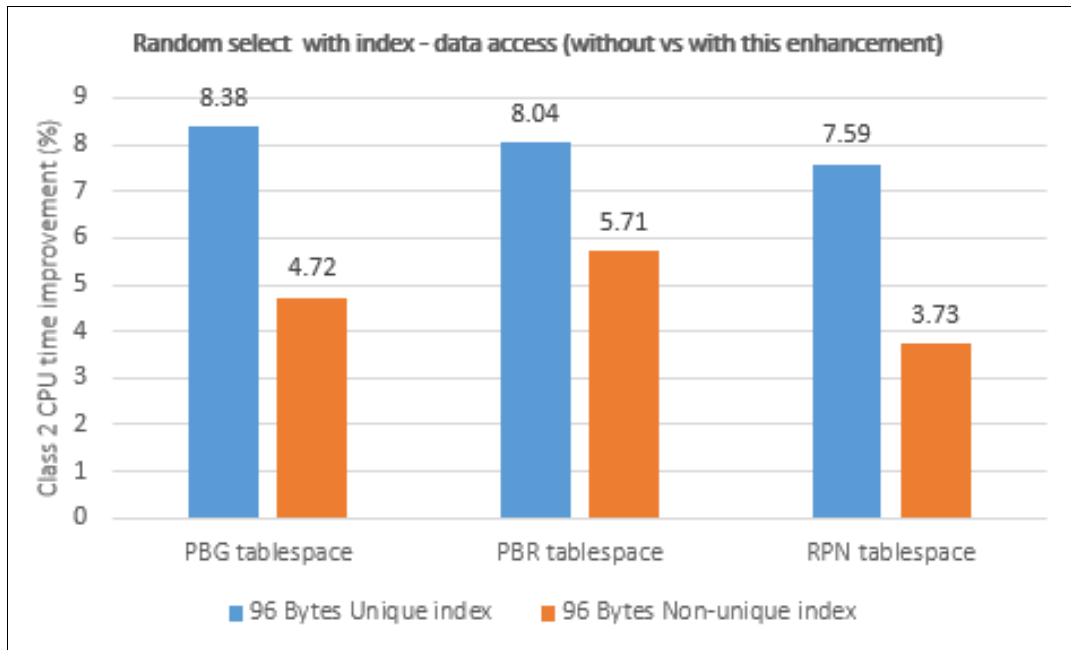


Figure 2-14 Class 2 CPU time improvement for random select with index-data access

For two-way data sharing for an index key size of greater than 64 bytes, the performance results for tests with a unique index with a key length of 96 bytes are shown in Figure 2-15 on page 35. Up to 13% CPU time improvement is observed for random index-data access, and there is no significant overhead for sequential insert, update, and delete operations.

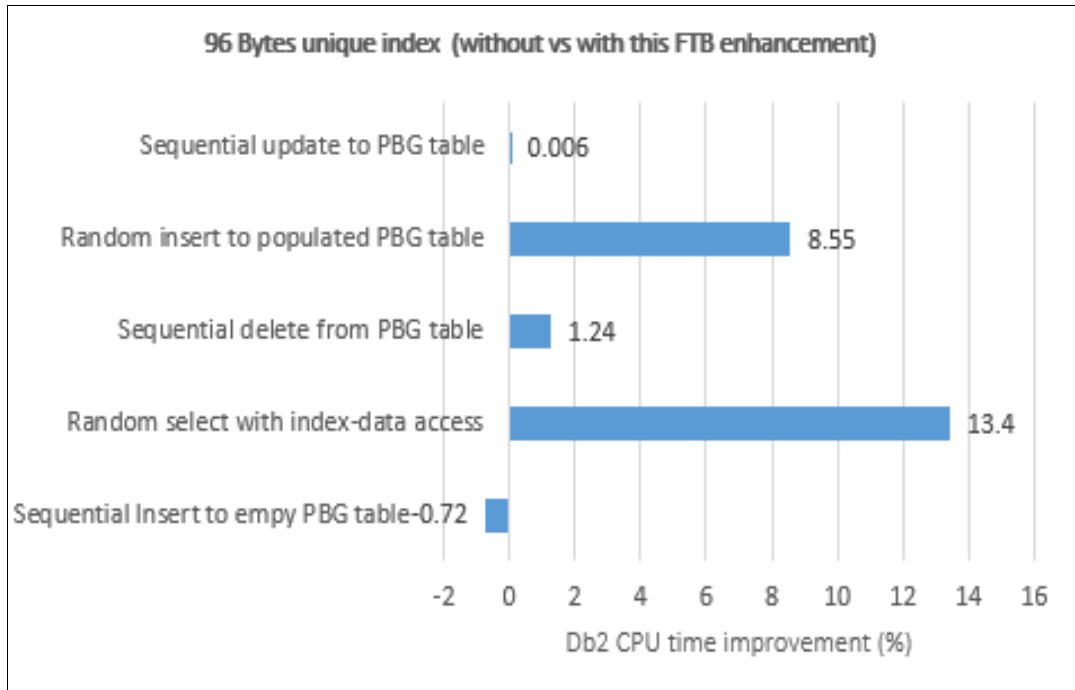


Figure 2-15 Db2 CPU time improvement for a 96-byte unique index

The performance results for a non-unique index are shown in Figure 2-16. Up to an 8% CPU time improvement is observed for random index-data access, and there is no significant overhead for sequential insert, update, and delete operations.

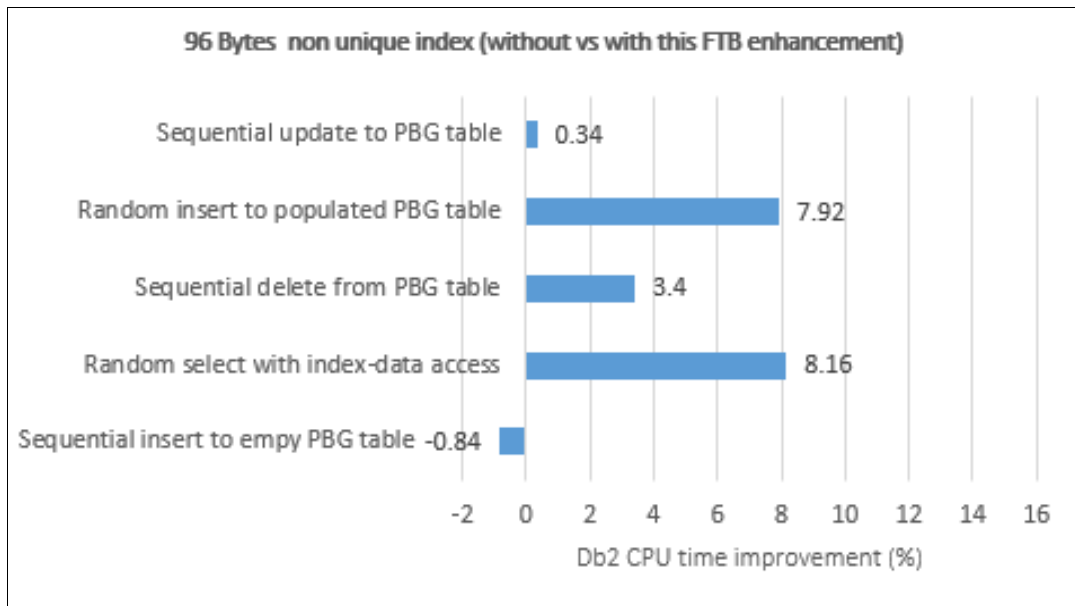


Figure 2-16 Db2 CPU time improvement for a 96-byte non-unique index

## Summary of results

This FTB enhancement extends the key size limit for eligible indexes. Up to 13% CPU time improvement was observed for random index-data access cases, and the enhancement introduced no noticeable performance degradation for other cases. Performance improvements are observed for all types of table spaces, from legacy classic partitioned table spaces to PBG table spaces and PBR table spaces.

### 2.6.3 Usage and monitoring

Db2 has been introducing usability and monitoring enhancements to the FTB function over the years.

For monitoring, new IFCID 2 fields and the new **-DISPLAY STATS(INDEXTRAVERSECOUNT)** command were introduced in Db2 12 with new-function APARs to help system administrators monitor and manage FTB more functions.

APAR PI72330 introduced six new FTB\_related fields for IFCID 2, as shown in Example 2-1.

#### *Example 2-1 New IFCID 2 FTB statistics fields*

---

QISTTRAVMIN - FTB threshold: minimum number of index traversals  
QISTFTBCANT - Total number of indexes that meet FTB criteria  
QISTFTBCAN - Total number of indexes that meet FTB criteria and the  
                  traverse count is above the threshold  
QISTFTBSIZE - Total memory allocation for all FTBs for this member  
                  (in MB)  
QISTFTBNUMP - Number of indexes for which FTB existed in the previous run  
                  of in-memory optimization  
QISTFTBNUMC - Number of indexes for which FTB exists in the current run  
                  of in-memory optimization

---

OMPE on z/OS 5.4.0 and later supports these new fields and prints them in the MISCELLANEOUS section of the Db2 statistics trace, as shown in Example 2-2.

#### *Example 2-2 OMPE formatted IFCID 2 FTB fields in statistics trace*

---

FTB THRESHOLD	1000.00
FTB CRITERIA MEET	518.00
FTB TRAVERSE ABOVE THE THRESHOLD	171.00
FTB TOTAL MEMORY ALLOCATION	49.00
FTB IN THE PREVIOUS OPTIMIZATION	136.00
FTB IN THE CURRENT OPTIMIZATION	136.00

---

APAR PH34859 introduces a new **-DISPLAY STATS(INDEXTRAVERSECOUNT)** command to display traverse counts of indexes, which can be abbreviated as **-DIS STATS(ITC)**.

This command helps database administrators to better manage FTB functions for individual indexes. For more information about how to use the command, see [-DISPLAY STATS \(Db2\)](#).

Example 2-3 on page 37 shows a sample command output with **DBNAME** specified. As the example shows, the indexes are ordered by traverse count with biggest one on top and the smallest at the bottom.

Example 2-3 The -DISPLAY STATS(ITC) output

---

```
DSNT830I  -CEB1
DBID  PSID  DBNAME  IX-SPACE  LVL  PART  TRAV.  COUNT
-----
00261 00198 TPCEB100 TESIX2   003 0001 0000138338
00261 00101 TPCEB100 TECIX1   003 0001 0000133485
00261 00054 TPCEB100 TETIX1   002 0971 0000113381
00261 00130 TPCEB100 TEBIX1   002 0001 0000095339
00261 00125 TPCEB100 TEADIX1  003 0001 0000082254
00261 00049 TPCEB100 TEZCIX1  002 0001 0000082254
00261 00176 TPCEB100 TENIIX1T 003 0001 0000082252
...
00261 00068 TPCEB100 TETRIX1  002 0310 0000000000
      -THRU                0325
00261 00096 TPCEB100 TESEIX1  003 0801 0000000000
00261 00096 TPCEB100 TESEIX1  003 0837 0000000000
00261 00096 TPCEB100 TESEIX1  003 0960 0000000000
***** DISPLAY OF STATS ENDED *****
```

---

To give Db2 system administrators more control over the usage of the FTB capability for specific indexes, Db2 12 APAR PH23238 introduced a new **SELECTED** option for the **INDEX\_MEMORY\_CONTROL** subsystem parameter. With this option, instead of letting Db2 automatically decide which indexes are suitable for using FTB structures for index access, only the indexes from SYSIBM.SYSINDEXCONTROL table rows with ACTION='A' (Automatic FTB creation) are considered for FTB eligibility. For more information about how to use the **SELECTED** option, see the [INDEX MEMORY CONTROL field](#).

How Db2 determines which indexes are suitable for FTB is usually effective. But if you decide to use the **SELECTED** option for the **INDEX\_MEMORY\_CONTROL** subsystem parameter, you can use the **-DISPLAY STATS(ITC)** command to get a better understanding of which indexes have the highest number of traverse counts. Choosing from those top indexes to insert into the SYSIBM.SYSINDEXCONTROL table with ACTION='A' is a good starting point.

Also, in Db2 12, APAR PH41751 eliminates message DSNI070I to reduce the number of console messages that are issued when fast index traversal (FTB) is used. The removed DSNI070I message reported changes to the number of objects that use FTB and other FTB-related status information for a Db2 subsystem. Customers reported seeing too many of these messages.

To display FTB storage usage, run the following command:

```
-DIS STATS(IMU) LIMIT(*)
```

To obtain detailed information about FTB usage in a Db2 subsystem, you can use IFCID 389 trace information. IFCID 389 also is part of statistics trace class 8. You also can issue the **-DISPLAY STATS** command with the **INDEXMEMORYUSAGE** option.

To check FTB eligibility when using Db2 12 at FL 501, issue the queries that are shown in Example 2-4.

*Example 2-4 FTB eligibility queries*

---

```
SELECT * FROM
(
  SELECT
    SUBSTR(A.CREATOR,1,10) AS TABLE_CREATOR,
    SUBSTR(A.NAME,1,10) AS TABLE_NAME,
    SUBSTR(B.CREATOR,1,10) AS INDEX_CREATOR,
    SUBSTR(B.NAME,1,10) AS INDEX_NAME,
    SUM(D.LENGTH
      + CASE D.COLTYPE
        WHEN 'VARBIN' THEN 2
        ELSE 0
      END
      + CASE D.NULLS
        WHEN 'Y' THEN 1
        ELSE 0
      END) AS INDEX_LENGTH
  FROM
    SYSIBM.SYSTABLES A, SYSIBM.SYSINDEXES B,
    SYSIBM.SYSKEYS C, SYSIBM.SYSCOLUMNS D
  WHERE A.NAME = B.TBNAME
    AND A.CREATOR = B.TBCREATOR
    AND B.CREATOR = C.IXCREATOR
    AND B.NAME = C.IXNAME
    AND A.NAME = D.TBNAME
    AND A.CREATOR = D.TBCREATOR
    AND C.COLNAME = D.NAME
    AND C.ORDERING <> ' '
    AND B.OLDEST_VERSION = B.CURRENT_VERSION
    AND D.COLTYPE <> 'TIMESTZ'
    AND B.DBID > 6
    AND B.UNIQUERULE NOT IN ( 'D','N')
  GROUP BY A.CREATOR,A.NAME, B.CREATOR, B.NAME
  ORDER BY A.CREATOR,A.NAME, B.CREATOR, B.NAME
) FTB_UNIQUE
WHERE FTB_UNIQUE.INDEX_LENGTH <= 128;

SELECT * FROM
(
  SELECT
    SUBSTR(A.CREATOR,1,10) AS TABLE_CREATOR,
    SUBSTR(A.NAME,1,10) AS TABLE_NAME,
    SUBSTR(B.CREATOR,1,10) AS INDEX_CREATOR,
    SUBSTR(B.NAME,1,10) AS INDEX_NAME,
    SUM(D.LENGTH
      + CASE D.COLTYPE
        WHEN 'VARCHAR' THEN 2
        WHEN 'VARBIN' THEN 4
        ELSE 0
      END
      + CASE D.NULLS
        WHEN 'Y' THEN 1
```

```

        ELSE 0
    END) AS INDEX_LENGTH
FROM
    SYSIBM.SYSTABLES A, SYSIBM.SYSINDEXES B,
    SYSIBM.SYSKEYS C, SYSIBM.SYSCOLUMNS D
WHERE A.NAME = B.TBNAME
    AND A.CREATOR = B.TBCREATOR
    AND B.CREATOR = C.IXCREATOR
    AND B.NAME = C.IXNAME
    AND A.NAME = D.TBNAME
    AND A.CREATOR = D.TBCREATOR
    AND C.COLNAME = D.NAME
    AND B.OLDEST_VERSION = B.CURRENT_VERSION
    AND D.COLTYPE <> 'TIMESTZ'
    AND B.DBID > 6
    AND B.UNIQUERULE = 'D'
GROUP BY A.CREATOR,A.NAME, B.CREATOR, B.NAME
ORDER BY A.CREATOR,A.NAME, B.CREATOR, B.NAME
) FTB_NON_UNIQUE
WHERE FTB_NON_UNIQUE.INDEX_LENGTH <= 120;

```

---

## 2.6.4 Conclusion

We hope that the information that is provided in this section conveys a strong message that the fast index traversal function is a powerful performance improvement feature. From its original introduction in 2016 at the GA of Db2 12, it has gradually matured and been improved over the years. With the expanded FTB capability to support non-unique indexes and indexes with longer key sizes, you can expect to see more indexes being accessed by using fast index traversal and enjoy more CPU cost savings.

## 2.7 DDF improvements

As in previous versions, Db2 13 also has a strong focus on providing performance enhancements in scalability. These enhancements help to optimize the performance of Db2 subsystems and reduce unplanned outages such as ones resulting from out of storage conditions. In the DDF area, Db2 13 provides the following storage-related scalability improvements:

- ▶ Changing DDF communication buffers to use DBAT thread storage
- ▶ Gradual DBAT contraction

### 2.7.1 Changing DDF communication buffers to use DBAT thread storage

When running with many concurrent threads, Db2 might encounter out-of-storage conditions for the single storage pool that is shared by all remote threads and used for communication I/O buffers. In systems with many concurrent threads that demand a large percentage of the single 2 GB storage pool that is used for communication buffers, you might hit storage pool exhaustion. This situation can cause a system outage and force you to reschedule or redistribute application processing to reduce the large concurrent demand for communication buffers as a workaround.

Db2 13 continues to use a single 2 GB storage pool for standard 8 K communication buffers. However, each thread now owns its own (up to) 2 GB storage pool, and this storage pool is deleted during DBAT termination. If the thread requires larger communication buffers, it allocates storage out of its thread-based storage pool. This enhancement is helpful for certain workloads that demand many communication buffers, such as connections that use large object (LOB) data or when Distributed Relational Database Architecture (DRDA) data encryption is enabled.

The storage relief from this enhancement greatly depends on the number of communications buffers that your workload needs. The performance measurements did not show any performance regression after this feature was added in Db2 13.

## 2.7.2 Gradual DBAT contraction

The Db2 DDF component manages its server DBATs. Basically, a DBAT is required to handle requests from a remote client, such as the following ones:

- ▶ Accepting a new connection
- ▶ Processing a transaction request
- ▶ Terminating the connection when requested by the client

While not processing a client request, a DBAT does not need to be attached to the client connection awaiting the next request from a client (CMTSTAT=INACTIVE). This capability allows Db2 to service many client connections with a relatively small population of DBAT server threads.

When a DBAT is not being used to service a connection request, the DBAT is “pooled” and considered disconnected from the connection. If a pooled DBAT is not used within the amount of time that is specified by the **POOLINAC** system parameter, it is terminated. A DBAT also is terminated at the end of processing after it is used cleanly to process up to 500 transactions. The **-STOP DDF MODE(SUSPEND)** command also terminates the disconnected pooled DBATs.

A high-performance DBAT also automatically terminates when it is used 500 times by its client connection. A high-performance DBAT also terminates if it does not receive a new transaction request within a **POOLINAC** amount of time. However, high-performance DBAT does not acknowledge a **POOLINAC** time of 0. If **POOLINAC** is set to 0, **POOLINAC** is assumed to be 120 seconds. Therefore, setting **POOLINAC** to 0 does not prevent Db2 from terminating a DBAT that was reused for over 500 requests.

When a (large) spike of incoming work over the network occurs, it requires Db2 to create many DBATs to handle the extra work. If the workload spike is short, there is a good chance that these additional DBATs are reused only a couple of times and mostly sit in the pool waiting to be reused. However, after being in the pool for more than the **POOLINAC** value, Db2 terminates those DBATs. Because the spike was short, many DBATs that are in the pool for longer than **POOLINAC** become eligible for termination concurrently.

When many DBATs terminate concurrently, a significant amount of system resources (including System Queue Area (SQA) storage) is required to handle these termination and deallocation requests. This situation might result in a significant system impact (RSMQ spin lock contention), or even a system outage (for example, if so much SQA storage is required that the SQA grows into ECSA, and you run out of ECSA as a result).

To provide some immediate relief, Db2 12 delivered PH36114 to reduce the number of DBAT terminations that occur concurrently.



Db2 13 further enhanced the DBAT reuse and termination process as follows:

- ▶ It increased the maximum number of DBAT transaction reuses to 500, up from 200, before a pooled disconnects or a high-performance DBAT terminates.
- ▶ Pooled, disconnected, or high-performance DBATs now terminate based on the **POOLINAC** subsystem parameter, plus an additional pseudo-random amount of time. This change ensures that fewer DBATs are likely to “expire” concurrently, avoiding spikes in several DBATs that are cleaned up concurrently.
- ▶ The Db2 DDF internal monitor that scans for time-based expiration is modified to run more efficiently. Pooled and disconnected threads, and DBATs are scanned every 15 seconds, and DBAT terminations are limited to a maximum of 50 per cycle. For high-performance DBATs, the DBAT scan is performed every 30 seconds. By increasing the frequency that the internal monitor runs, fewer DBATs are expected to be concurrently terminated per monitor cycle.

Db2 thread storage contraction was improved. With these changes, which are intended to improve DBAT termination behavior in Db2, you see a reduction in the overall frequency and number of DBAT terminations, which also helps to reduce the number of concurrent DBAT terminations that are often caused by a short workload spike.

### Performance measurement

To evaluate the new DBAT termination behavior, a client workload with 2000 threads that are connected to the Db2 subsystem is used. After the workload runs stable for 2 minutes, all threads of this workload stop performing SQL work while they are still connected to the Db2 subsystem. To analyze the DBAT termination behavior, we use the ACTIVE\_DBATS metric. It is the total number of allocated DBATs (in use or in the pool).

Figure 2-17 shows the ACTIVE\_DBATS metric from the Db2 statistics trace. The data shows the number of active DBATs gradually decreasing at a rate of 50 DBAT termination per cycle, as expected.

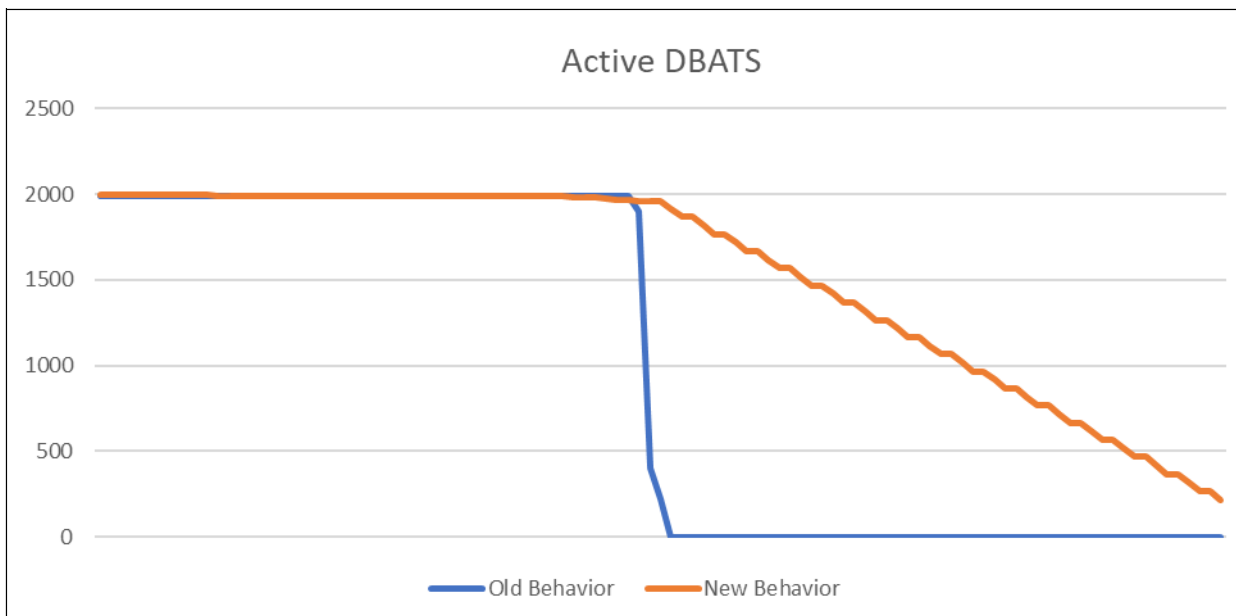


Figure 2-17 Gradual DBAT contraction

## **Conclusion**

The DDF improvements in DBAT termination behavior in Db2 12 and Db2 13 reduce the frequency and number of concurrent DBAT terminations that are caused by a short-term DBAT usage spike, and they reduce the risk of system outages due to scalability constraints.



## Synergy with the IBM Z platform

As with previous Db2 versions, Db2 13 for z/OS leverages the latest improvements that are available on the IBM Z platform. Db2 13 increases its synergy with IBM zSystems hardware and IBM Z software to provide better performance, stronger resilience, and better functions for overall improved value.

This chapter includes the following topics:

- ▶ IBM z16
- ▶ Data-sharing overhead reduction with short reach coupling links
- ▶ CFLEVEL 25 and Db2 CF structures
- ▶ GBP residency time
- ▶ System Recovery Boost: IBM z15 and later
- ▶ REORG table space with variable length records on IBM z15 with IBM Z Sort
- ▶ Leveraging Z Sort by SQL sort
- ▶ Large object compression
- ▶ Asynchronous cross-invalidation: IBM z14 and later
- ▶ zHyperLink I/O
- ▶ Huffman compression for data
- ▶ Encrypting Db2 data with z/OS data set encryption and CF structure encryption

## 3.1 IBM z16

IBM z16 brings artificial intelligence (AI), scalability, and cyber resiliency to your hybrid cloud. IBM z16 introduces the following key features that Db2 leverages explicitly or implicitly:

### **Single instruction, multiple data (SIMD) and on-chip AI accelerator**

SQL Data Insights in Db2 13 leverages improved AI capability inside IBM z16 through the IBM Z AI Optimization Library (zAIO) to leverage AI optimization between SIMD and the on-chip accelerator based on the request.

### **Group buffer pool (GBP) residency time**

CFCC level 25 is delivered on the IBM z16 and reports the cache residency time of both data and directory entries. The residency time can be used to tune the size of GBP.

### **Coupling Facility (CF) scalability and coupling short link improvement**

IBM z16 provides two important CF-related improvements. IBM z16 CF images can scale better with many CPs than CF images on IBM z15. Additionally, performance improvements when using CF short link (CS5) can reduce CPU time for Db2 data-sharing configurations.

### **System Recovery Boost (SRB) enhancements**

IBM z16 supports the middleware restart boost in addition to initial program load (IPL) boost. Db2 can leverage the middleware restart boost.

### 3.1.1 Db2 workload performance with IBM z16

IBM z16 performance was evaluated by using various types of Db2 workloads to understand the performance characteristics. The measurements were conducted by using equivalent hardware configurations and software maintenance levels, other than switching the processor models.

#### **Measurement environment**

To conduct the measurements, we used Db2 13 for z/OS and the same level of z/OS level, either z/OS 2.4 or z/OS 2.5. The Db2 CPU time, including both general-purpose processors and IBM zSystems Integrated Information Processors (zIIPs), was the comparison point.

Figure 3-1 on page 45 shows the percentage of CPU time improvement running on IBM z16 compared to IBM z15. The workloads that are used here are described in Appendix A, “IBM Db2 workloads” on page 403.

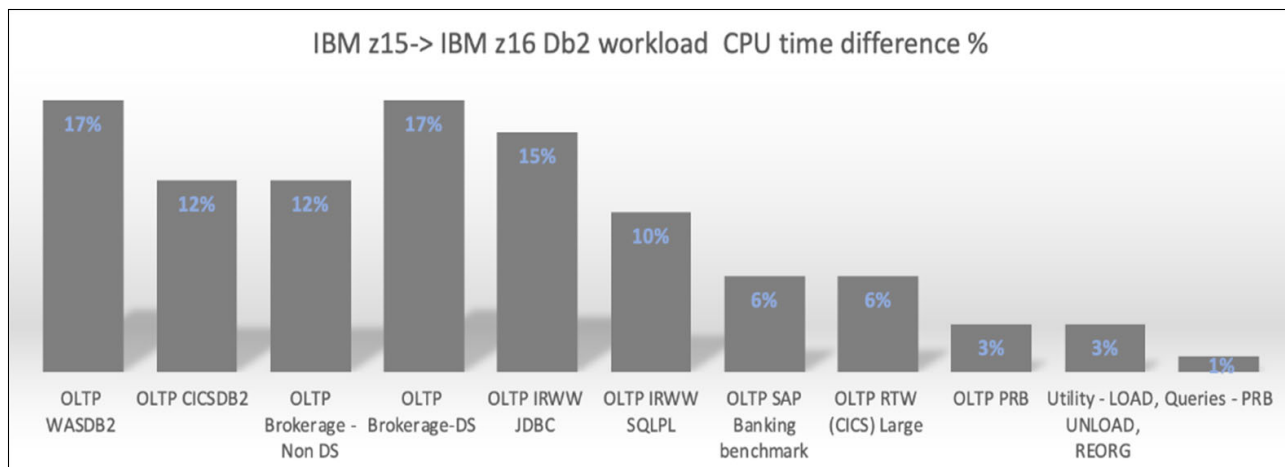


Figure 3-1 Percentage of Db2 CPU time difference between IBM z15 and IBM z16

### Summary of results and considerations

The improvements that were observed with Db2 workloads running on IBM z16 compared to IBM z15 vary considerably between 17% and almost as equivalent, as shown in Figure 3-1. In the measurements, the workloads with many concurrent threads show more improvement compared to workload with less concurrency, such as Db2 batch workloads, utilities, or complex queries.

The analysis indicates that the wide range of variability is due to the difference in the processor cache-hit behavior. IBM z16 processors use 7-nm chip technology with an updated cache hierarchy on the central processor complex (CPC) drawer. IBM z16 has a larger L2 cache than IBM z15, with up to 32 MB of virtual private cache. The L3 and L4 cache are using a virtual shared cache on the IBM z16.

In general, workload performance is sensitive to how deep into the memory hierarchy the processor must go to retrieve the instructions and data to run the workloads. This characteristic is described as relative nest intensity (RNI).

The workloads that mostly use the L2 cache can be characterized as low-RNI workloads. They are typically Db2 batches or queries. Conversely, workloads that require a deeper level of cache hierarchy are called high-RNI workloads. Typically, transactional workloads with many concurrent threads with transaction managers running on z/OS fall into this category. Due to the cache hierarchy updates, a workload with high RNI might see more improvement compared to a low-RNI workload that runs on IBM z16.

For more information about the cache level differences, see *IBM z16 (3931) Technical Guide*, SG24-8951.

We also observed more improvement for the IBM brokerage transactional workload running in a data-sharing configuration compared to a non-data-sharing configuration. This difference is attributed to the performance optimization of the IBM z16 coupling link latency. For more information about this improvement, see 3.2, “Data-sharing overhead reduction with short reach coupling links” on page 46.

## 3.2 Data-sharing overhead reduction with short reach coupling links

CF CFLEVEL 25, which comes with IBM z16, improves “short reach coupling” (CS5) latency with the following enhancements:

- ▶ Removes the memory round trip to retrieve message command blocks.
- ▶ Removes the cross-fiber handshake to send data for a CF write command.

These improvements result in improved synchronous CF request service times, which reduce Db2 data sharing overhead.

### 3.2.1 Requirements

To leverage these improvements, you must be using IBM z16 with CFLEVEL 25 or later. IBM z15 running CFLEVEL 24 running the same workload is used as a basis for comparison.

No specific Db2 version or function level (FL) is required for a data-sharing group to leverage the benefits of this z16 enhancement. If there are Db2 CF structures on CFs at CFLEVEL 25 and connected to the z/OS logical partitions (LPARs) (at z/OS 2.3 or later with PTFs for APAR OA60650 applied) through CS5 type CF links, there should be data-sharing processing savings.

### 3.2.2 Performance measurement

To evaluate this enhancement, the following performance study was conducted.

#### Measurement environment

Two sysplexes are employed: one on IBM z15 hardware, and the other on IBM z16. Each sysplex consists of two Internal Coupling Facilities (ICFs) and two LPARs. Separate CS5 links are used to link LPARs and ICFs. Another separate CS5 link also connects the two ICFs. The tests use asynchronous duplexing for the Db2 lock structure and synchronous duplexing of the shared communications area (SCA) structure. All GBPs are duplexed with the structures that are spread over the two ICFs.

The IBM brokerage transaction workload is the workhorse that is used to evaluate this feature. The workload is driven by a Linux on IBM Z driver, which is outside of the sysplex but within the same IBM z15 or IBM z16 box as the sysplex.

#### Measurement scenario description

Two sets of performance measurements were collected: one set is run on IBM z15 with CFLEVEL 24, and the other set is run on IBM z16 with CFLEVL 25. Each set consists of two measurements: non-data sharing and two-way data sharing. Comparing the non-data-sharing measurement against the data-sharing run, the data-sharing overhead is determined for the IBM z15. This measurement serves as the baseline. Running the same measurements on IBM z16 and calculating the data-sharing overhead, a reduction of the data-sharing overhead is expected.

#### Results

Table 3-1 on page 47 highlights data from each collected measurement.

Table 3-1 Two-way data-sharing overhead for IBM z15 and IBM z16

Item	IBM z15 (CFLEVEL 24)	IBM z16 (CFLEVEL 25)	Delta = IBM z16 / IBM z15
<b>Non-data sharing</b>			
Total Db2 CPU time (ms/commit) (class 2 + MSTR + DBM1 + internal resource lock manager (IRLM))	0.596	0.524	12% better in IBM z16
Internal throughput rate (ITR) (commits/sec.)	9482.609	10835.091	14% better in IBM z16
<b>Two-way data sharing</b>			
Group scope: Total Db2 CPU time (ms/commit) (class 2 + MSTR + DBM1 + IRLM)	0.912	0.758	17% better in IBM z16
Group scope: ITR (commits/sec.)	12891.666	15417.941	20% better in IBM z16
Two-way data-sharing overhead	32.0%	28.8%	3.2 percentage point reduction in z16

The two-way data-sharing overhead is calculated as follows:

Two-way data-sharing overhead =  $1 - (\text{ITR of member}_1 + \text{ITR of member}_2) / (2 * \text{ITR of non-data-sharing member})$

As indicated in Table 3-1, using IBM z16 along with CFLEVEL 25 shaves off 3.2 percentage points of the (two-way) data-sharing overhead compared to IBM z15 that uses CFLEVEL 24.

Looking at the RMF CF activity reports, the average SYNC CF service time for IBM z15 is 5.75 microseconds, and it is 3.85 microseconds for IBM z16, which is an improvement of 33% though both IBM z15 and IBM z16 processors are running at a 5.2 GHz clock speed.

CF access accounts for most, if not all, of the Db2 data-sharing cost. CF synchronous access times are directly related to CPU time, so enhancing CF SYNC access time directly reduces the data-sharing cost.

In this evaluation, the total CF SYNC time in the two-way data-sharing measurement is 484 IBM z15 CPU seconds and 336 IBM z16 CPU seconds, a reduction of 30%, which is in line with the average SYNC CF service time difference.

Because the data-sharing cost is much cheaper on IBM z16 than IBM z15, the data-sharing measurement (CPU time or ITR) improves more than non-data sharing on IBM z16 compared to IBM z15.

### 3.2.3 Monitoring information

Db2 Accounting and Statistics traces along with RMF data are used to monitor system performance.

### 3.2.4 Conclusion

For the IBM brokerage transaction workload, the two-way data-sharing overhead is reduced by 3.2 percentage points. This reduction is a direct result of the improved CPU-synchronous service times, from 5.75 microseconds per SYNC access on IBM z15 to 3.85 on IBM z16, a 33% reduction.

However, the performance of Internal Coupling (ICP) links or Long Reach Coupling (CL5) links are not enhanced in CFLEVEL 25.

## 3.3 CFLEVEL 25 and Db2 CF structures

The IBM brokerage workload is designed to run on the IBM z16 and use CFLEVEL 25 for the ICFs. Before the measurements could be conducted, we needed to migrate the workload from the IBM z15 machine to the IBM z16 machine. At first, we copied the CFRM policy from the CFLEVEL 24 (IBM z15) machine to the IBM z16 without resizing the CF structures. In doing so, we made some interesting observations worth sharing.

We want to emphasize that this section describes our experiences with migrating the IBM brokerage workload from CFLEVEL 24 (IBM z15) to 25 (IBM z16). We make this information available so that you can avoid the same issues that we experienced. It is not intended to provide any technical advice that the Db2 community must follow.

The following excerpt, which is taken from the section titled *CF structure sizing increases for CFLEVEL 25* in the Family 3931+01 IBM z16 announcement letter, provides the official advice for CFLEVEL 25:

“CF structure memory size requirements often increase when migrating to a higher CFLEVEL. IBM z16 CFLEVEL 25 is no exception to this general rule, and in fact the structure size increases might be more noticeable going to CFLEVEL 25 than for some previous CFLEVEL migrations, particularly for structures whose absolute size is small (for example, 100 MB or less). Clients are urged to carefully resize their CF structures as part of migrating to CFLEVEL 25.”

Our experience is a result of having a small SCA structure for the workload.

### 3.3.1 Measurement environment

The IBM brokerage workload that is used during this evaluation was run in the following environment:

- ▶ An IBM z16 sysplex consisting of two MVS LPARs
- ▶ ICFs running CFLEVEL 25
- ▶ z/OS 2.5
- ▶ Db2 13 at FL 501

### 3.3.2 Results

The Db2 SCA list structure was defined with a SIZE setting of 10M when using CFLEVEL 24. When using CFLEVEL 25, Db2 at FL 501 failed to start with a 10M size SCA.



Messages DSN7505A and DSN7512A were issued and indicated that the 10M size was too small. Therefore, we had to enlarge the SCA structure size to 12M for the Db2 members to restart successfully on the IBM z16 machine.

The SCA structure has a different number of directory entries and data elements that are available in CFLEVEL 25 compared to CFLEVEL 24. Table 3-2 shows the allocated size, number of directory entries, and number of data elements of the SCA structures that we used under CFLEVEL 24 and 25.

*Table 3-2 Db2 SCA structure geometry between CFLEVEL 25 and 24*

List structure	CFLEVEL 24	CFLEVEL 25	CFLEVEL 25 versus 24
DSNCEB0_SCA			
Actual size (KB)	10,240	12,288	20% more
Number of entries	5,695	3,296	42% less
Number of elements	11,324	5,918	47% less

As shown in Table 3-2, the SCA structure uses 20% more storage in CFLEVEL 25 than 24, yet both the number of directory entries and data elements are fewer than CFLEVEL 24. Because we used a small SCA structure for CFLEVEL 24, we were unlucky that without making it larger on CFLEVEL 25, there was not enough space in the structure for Db2 to restart successfully, and this situation drew our attention to the memory size requirement increase for CFLEVEL 25.

Other data-sharing workloads with larger SCA structure sizes, such as the relational transaction workload (RTW) workload, which uses a SIZE (80M) SCA structure, did not encounter the same SCA storage shortage issue after migrating to CFLEVEL 25 with the same structure size.

We also examined the GBP structures to understand whether the number of directory entries and data elements change after migrating to CFLEVEL 25 without structure size changes.

Table 3-3 illustrates the allocated size, number of directory entries, and number of data elements of two Db2 GBP structures for the 4 K local buffer pools for CFLEVEL 24 and 25.

Table 3-3 Db2 GBP structure for 4 K buffer pools: CFLEVEL 25 versus CFLEVEL 24

Cache structure	CFLEVEL 24	CFLEVEL 25	CFLEVEL 25 versus 24
DSNCEB0_GBP0			
Actual size (KB) DSNB758I	64,512	64,512	Same
Number of entries DSNB759I	51,448	51,252	0.4% fewer
Number of elements DSNB759I	10,289	10,248	0.4% fewer
DSNCEB0_GBP8			
Actual size (KB) DSNB758I	27,262,976	27,262,976	Same
Number of entries DSNB759I	23,074,952	22,772,892	1.3% fewer
Number of elements DSNB759I	4,614,989	4,554,578	1.3% fewer

DSNCEB0\_GBP0 is the smallest GBP (63M in size), and GBP8 is the largest (26G in size) that corresponds to 4 K local buffer pool BP8. The shrinkage in directory entries and data elements, from CFLEVEL 24 to 25, is 0.4% - 1.3% for the IBM brokerage workload. On CFLEVEL 24, we do not see any directory reclaims for the GBP structures. However, with CFLEVEL 25, because of reductions in total available directory entries, we see a small amount of directory reclaims for some of the GBP structures, although the small amount of directory reclaims does not seem to impact performance.

Although most local buffer pools are 4 K size buffer pools, there is one 32 K local buffer pool that requires a corresponding GBP structure DSNCEB0\_GBP32K. Table 3-4 illustrates the GBP32K comparison between CFLEVEL 25 and 24.

Table 3-4 Db2 GBP structure for 32 K buffer pools; CFLEVEL 25 versus CFLEVEL 24

Cache structure	CFLEVEL 24	CFLEVEL 25	CFLEVEL 25 versus 24
DSNCEB0_GBP32K			
Actual size (KB) DSNB758I	16,384	16,384	Same
Number of entries DSNB759I	1,434	1,045	27% fewer
Number of elements DSNB759I	285	208	27% fewer

With the same allocated size for GBP32K, both the number of directory entries and data elements are 27% fewer in CFLEVEL 25 than in CFLEVEL 24. This difference might impact system performance if GBP32K is heavily accessed.

No 8 K nor 16 K GBPs are allocated in this workload. Therefore, no information can be referenced.

Table 3-5 shows the DSNCEB0\_LOCK1 comparison between CFLEVEL 24 and 25.

Table 3-5 Db2 lock structures: CFLEVEL 25 versus CFLEVEL 24

Lock structure	CFLEVEL 24	CFLEVEL 25	CFLEVEL 25 versus 24
DSNCEB0_LOCK1			
Actual size (KB)	2,097,152	2,097,152	Same
Number of entries	3,597,609	3,596,548	Comparable
Locks	134,217,728	134,217,728	Same
DPLXCN Transport Layer Security (TLS)	4,096	4,096	Same

On the positive side, the geometry of the Db2 lock structure remains constant between CFLEVEL 24 and 25 for the IBM brokerage workload.

### Summary of results

Although this study is not an exhaustive or comprehensive evaluation of CFLEVEL 25, it did reveal the fact that when the IBM brokerage data-sharing workload was moved to an IBM z16 machine, and after applying the same CFRM policy settings that were running fine on CFLEVEL 24 CFs to the CFLEVEL 25 CFs, we observed a significant drop in the number of “entries” that can be stored in the SCA structure and some of the GBP structures with relatively small sizes.

**Note:** Re-evaluate your CF structures sizes before allocating them in a CF running CFLEVEL 25 and later. For official IBM CFLEVEL25 advice, see “CF structure sizing increases for CFLEVEL 25” in the [Family 3931+01 IBM z16](#) announcement letter.

## 3.4 GBP residency time

With this enhancement, the CF provides statistics about the GBP residency time for data and directory entries, which indicate how long data and directory entries are in the GBP before they are reclaimed. This information can help to improve GBP sizing, which results in improved data-sharing performance.

Two new statistics are added to the GBP statistics storage areas: One is the average time that a data element stays in a GBP before it is reclaimed, and the other is the average time that a directory entry is in a GBP before it is reclaimed. The values are returned to Db2 from the CF through z/OS. They are moving weighted averages that approximate the residency time in microseconds averaged on a per-cache-storage-class basis.

These two metrics can be obtained through the Instrumentation Facility Component Identifier (IFCID) 230, 254 record traces, or by issuing the **-DISPLAY GROUPBUFFERPOOL** command with the **GDETAIL** option. For more information about this enhancement, see 12.4, “Group buffer pool residency time enhancements” on page 356.

For this feature to work, Db2 must be running on an IBM z16 LPAR with z/OS 2.4 or later that has the necessary PTFs for APAR [OA60650](#) to support the cache residency time metrics. Also, the CF must be running at CFLEVEL 25 or later.

### 3.4.1 Performance measurement

The Db2 for z/OS team performed a series of two-way data-sharing measurements to understand the performance effect that the GBP residency time has on data-sharing efficiency.

#### Measurement environment

The IBM brokerage workload that is used during this evaluation was run in the following environment:

- ▶ An IBM z16 sysplex consisting of two MVS LPARs
- ▶ Two ICFs running CFLEVEL 25
- ▶ z/OS 2.5
- ▶ Db2 13 at FL 501

#### Results

A series of nine two-way data-sharing measurements were collected to understand the performance effect of the GBP residency time for a data-sharing environment. In particular, the size of four GBPs was gradually increased to observe the changes in their data and directory residency times along with the data-sharing group's ITR, which is a yardstick for system performance.

The nine measurements were designated as state I - IX in Table 3-6. Each state used different GBP sizes for GBP12, GBP21, GBP23, and GBP24.

Table 3-6 States I - IX for four varying GBP sizes

State	GBP12 size (GB)	GBP21 size (GB)	GBP23 size (GB)	GBP24 size (GB)
I	0.25	1	1	0.25
II	0.50	2	2	0.50
III	1	4	4	1
IV	2	8	8	2
V	3	12	12	3
VI	5	17	17	5
VII	20	17	17	20
VIII	20	40	40	20
IX	20	80	80	20

Figure 3-2 on page 53 through Figure 3-5 on page 54 depict the GBP residency time for data and directory entries at the end of each measurement for the respective GBPs.

**Note:** If there are no data or directory reclaims for a GBP structure after the structure is allocated, the residency time that is returned is 0. The residency time is calculated after a cache structure is allocated. The current allocation's residency time is not affected by previous allocations.

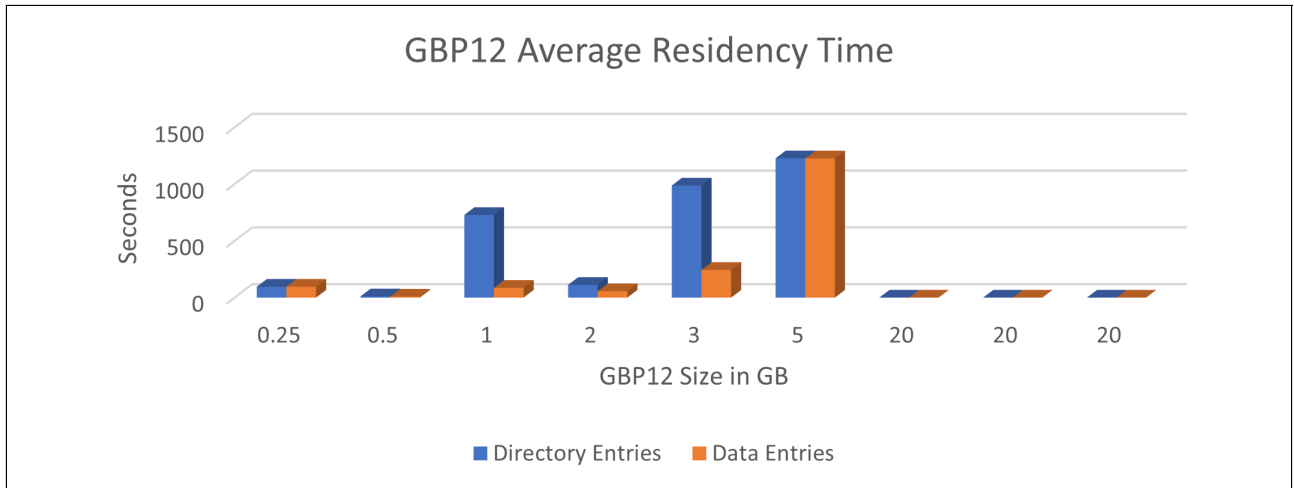


Figure 3-2 GBP12: Average residency time

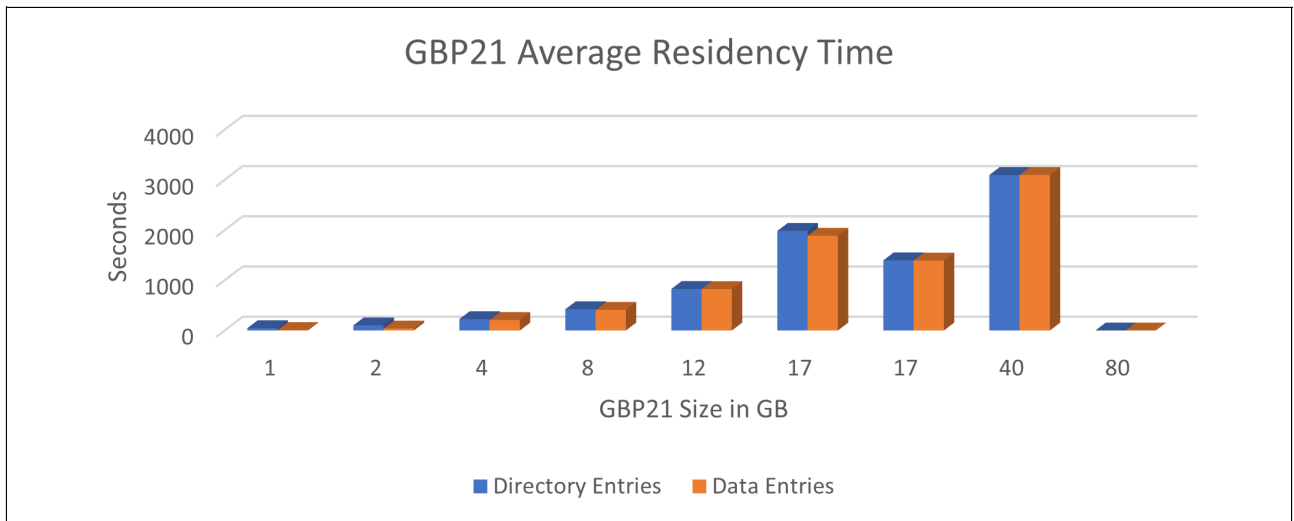


Figure 3-3 GBP21: Average residency time

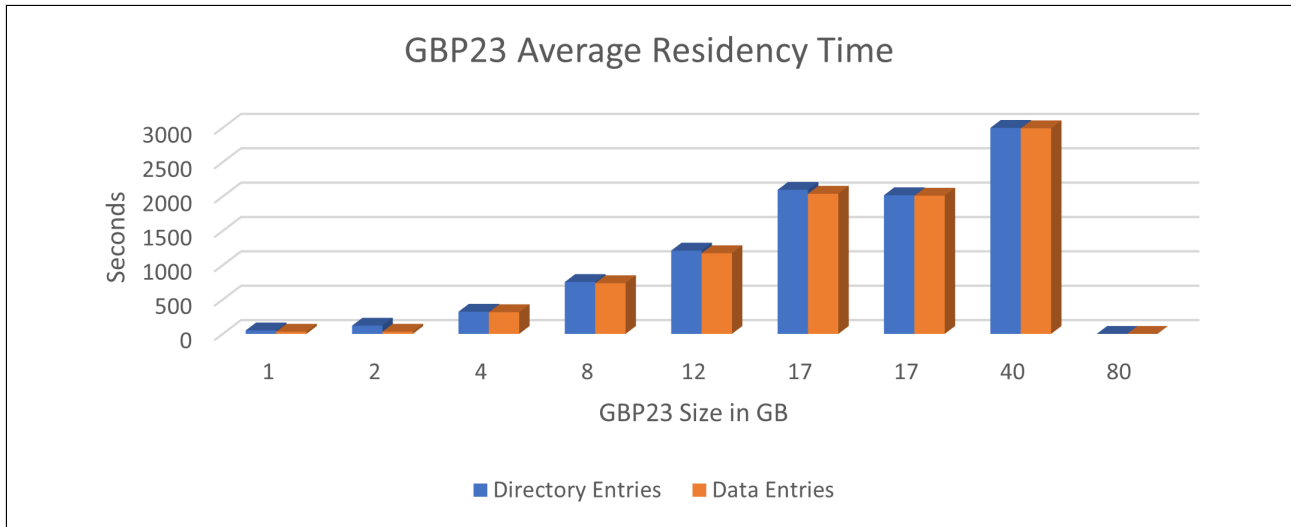


Figure 3-4 GBP23: Average residency time

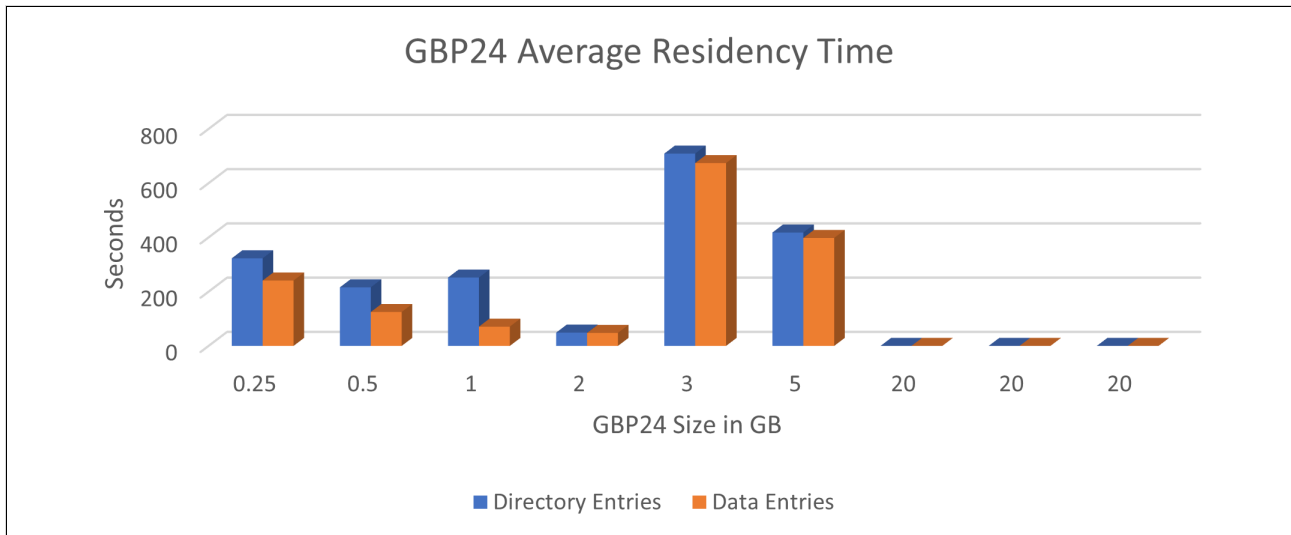


Figure 3-5 GBP24: Average residency time

The Db2 data-sharing group ITR for states I - IX are shown in Figure 3-6 on page 55.

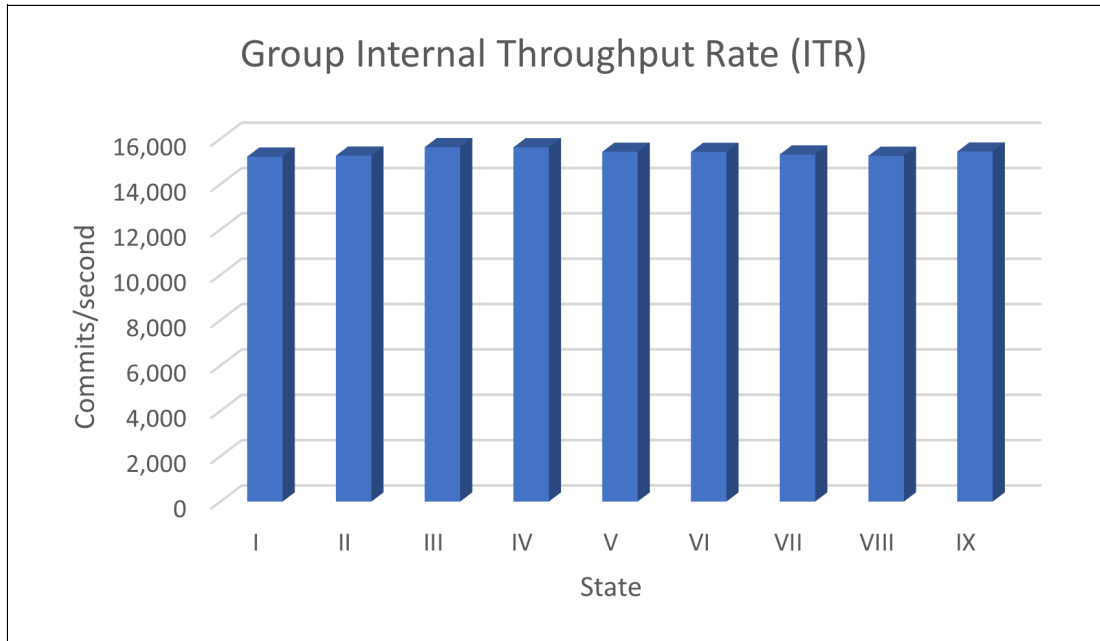


Figure 3-6 Db2 group ITR for various states

The distribution of the group's ITR is 15,200 - 15,700 commits per second, which is a mere 3.3% difference and barely above the measurement noise level. However, it is surprising to see that the group ITR of state IX is not the best one. With zero GBP residency time in both data and directory entries, which means no reclaim is performed for any of the GBP structures, we expected state IX to be among the best performing scenarios. Further investigations were conducted to better understand this unexpected behavior.

First, the request rate to the CF, where the primary GBPs are during measurement interval, was examined and plotted in Figure 3-7. It turns out that the request rate for states VIII and IX is approximately 4 - 8% higher than the average request rate of nine measurements, which means that the CF, where the primary GBPs are, has more work to do, which leads to higher CF utilization, as shown in Figure 3-8, than the other states.

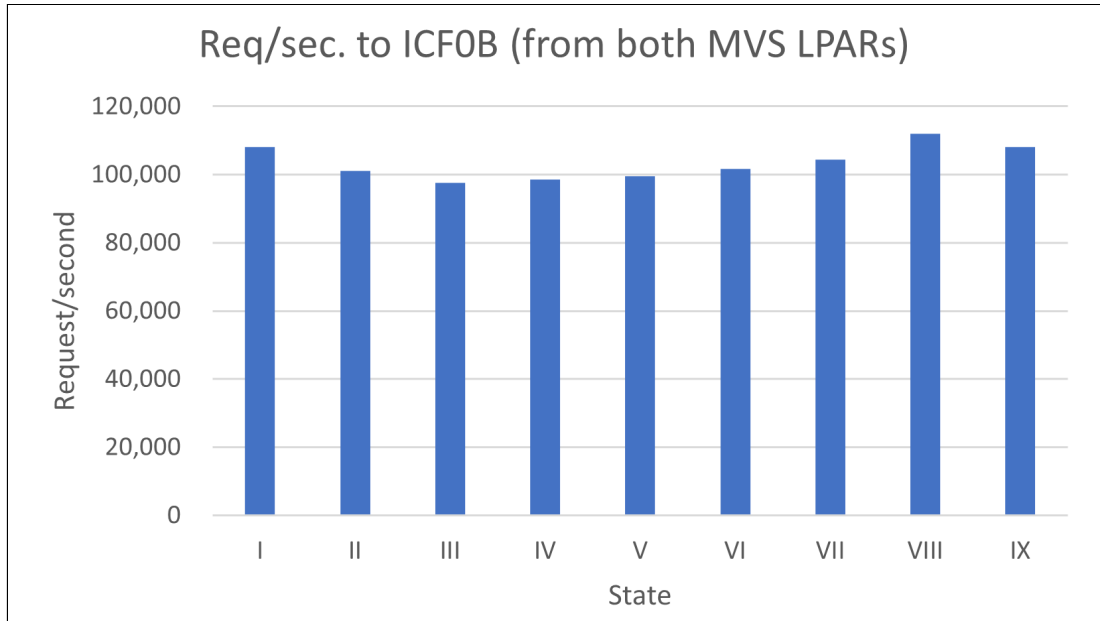


Figure 3-7 Request rate to CF where primary GBPs are

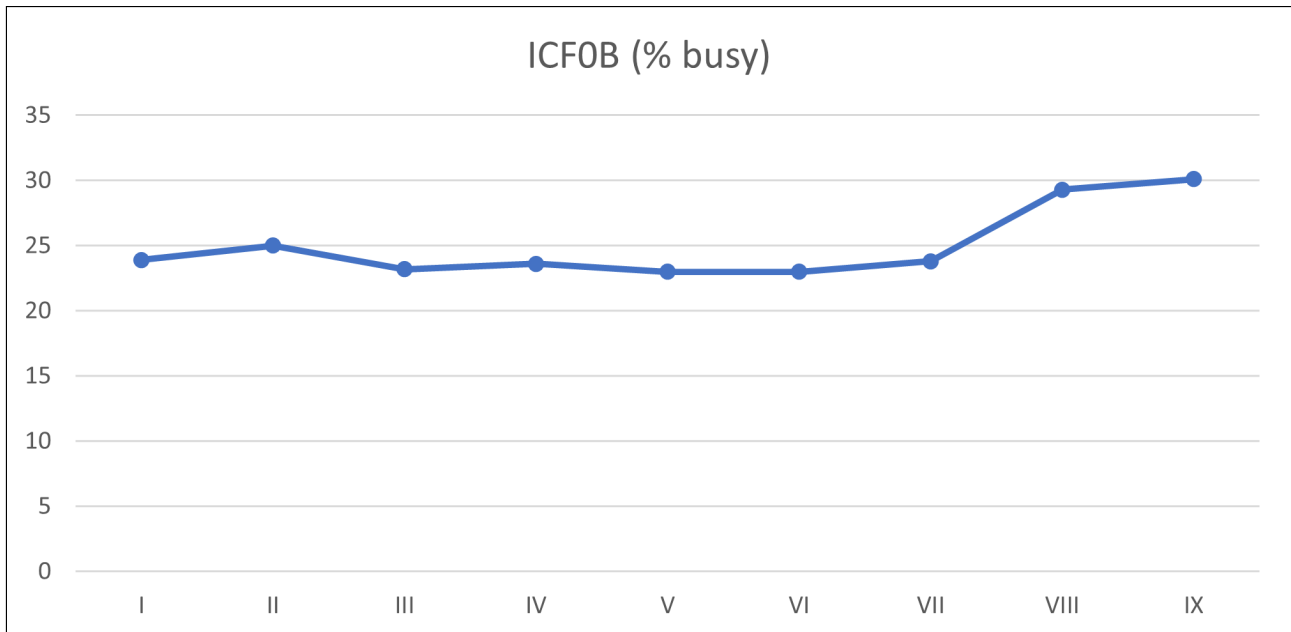


Figure 3-8 CF utilization of the CF with primary GBPs across nine states

It is evident that the CF utilization of states VIII and IX is much higher than the other measurements, and later CF utilization leads to longer CF SYNC service times.



The CF SYNC service times for the four GBPs across states I - IX are extracted from CF activity report and plotted in Figure 3-9 through Figure 3-12 on page 58.

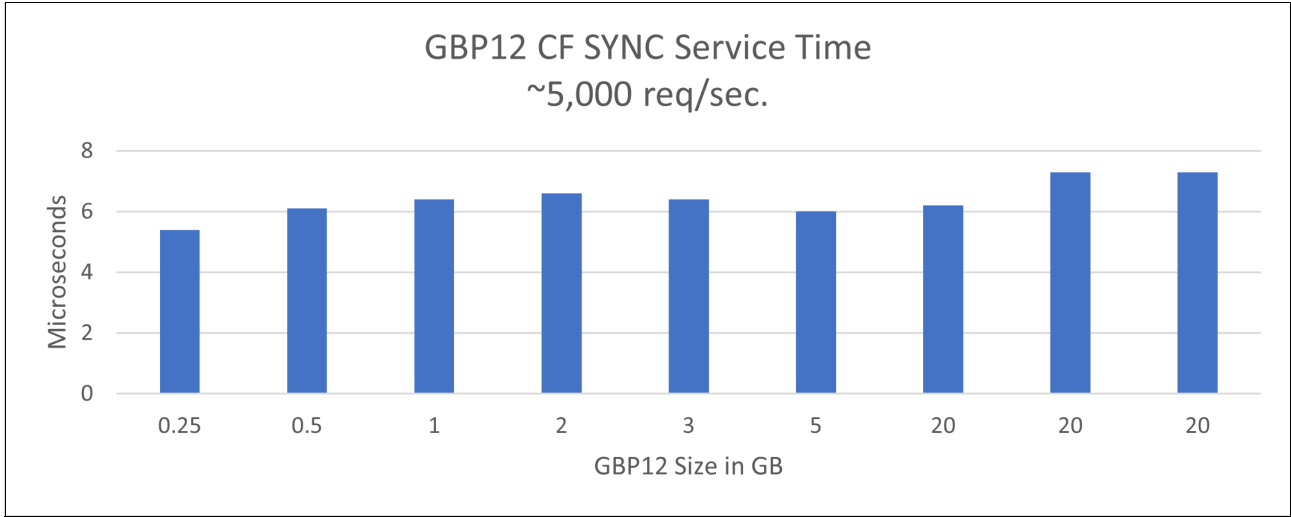


Figure 3-9 GBP12: CF SYNC service time

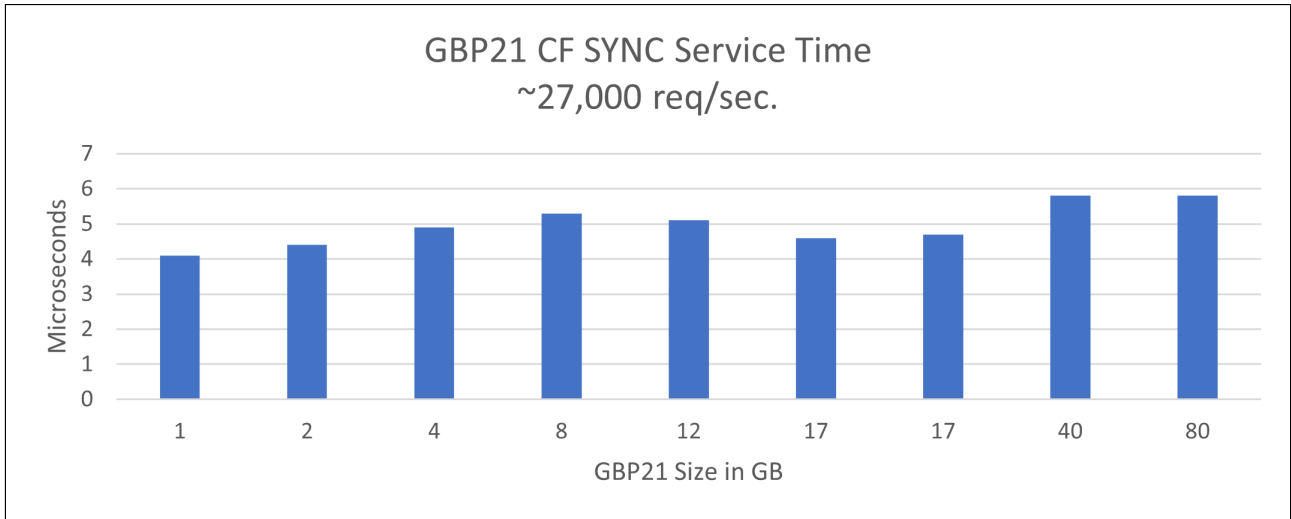


Figure 3-10 GBP21: CF SYNC service time

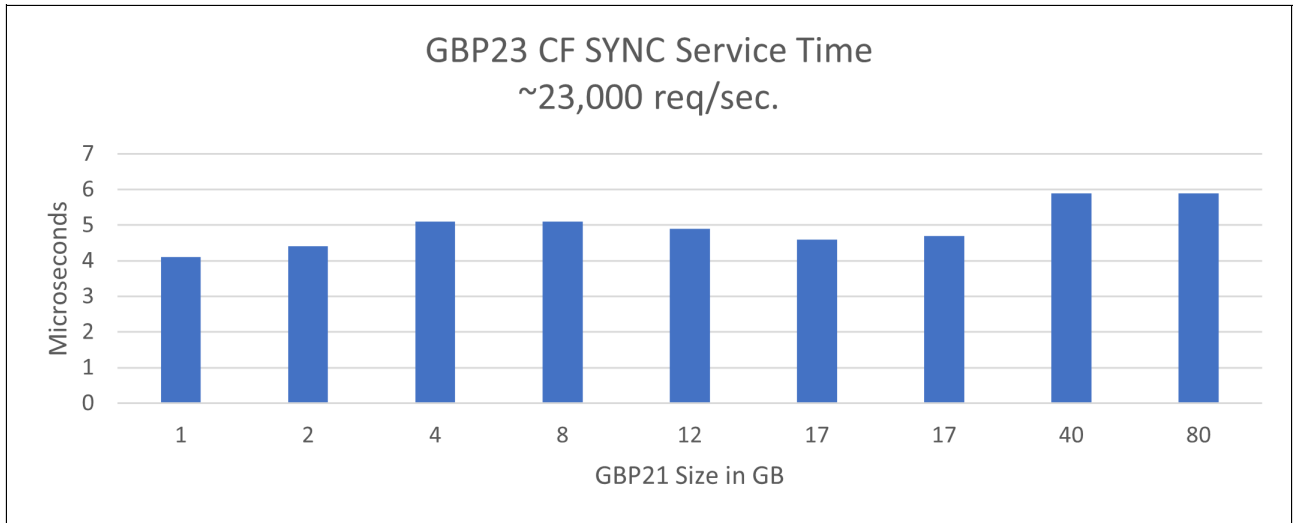


Figure 3-11 GBP23: CF SYNC service time

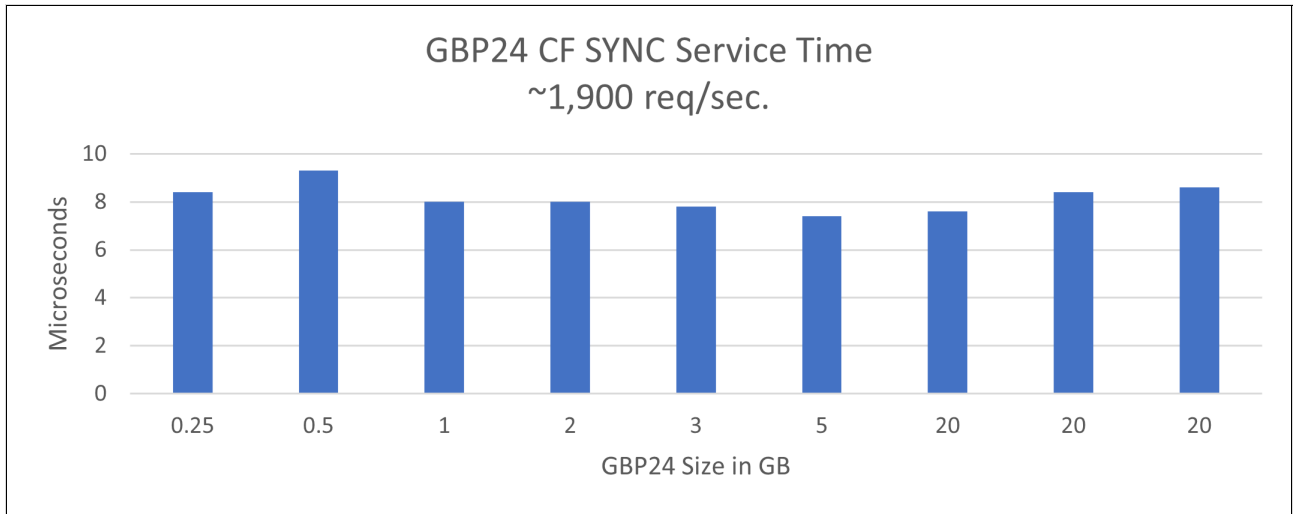


Figure 3-12 GBP24: CF SYNC service time

Figure 3-9 on page 57 through Figure 3-12 verify that more CF requests lead to higher CF utilization, which results in longer CF SYNC service times.

Armed with this information, we proceeded to calculate the total group's CF SYNC service time for each state during the measurement periods as a sum (GBPxx CF SYNC service time per request x total SYNC requests during the measurement duration). Figure 3-13 on page 59 shows the outcome.

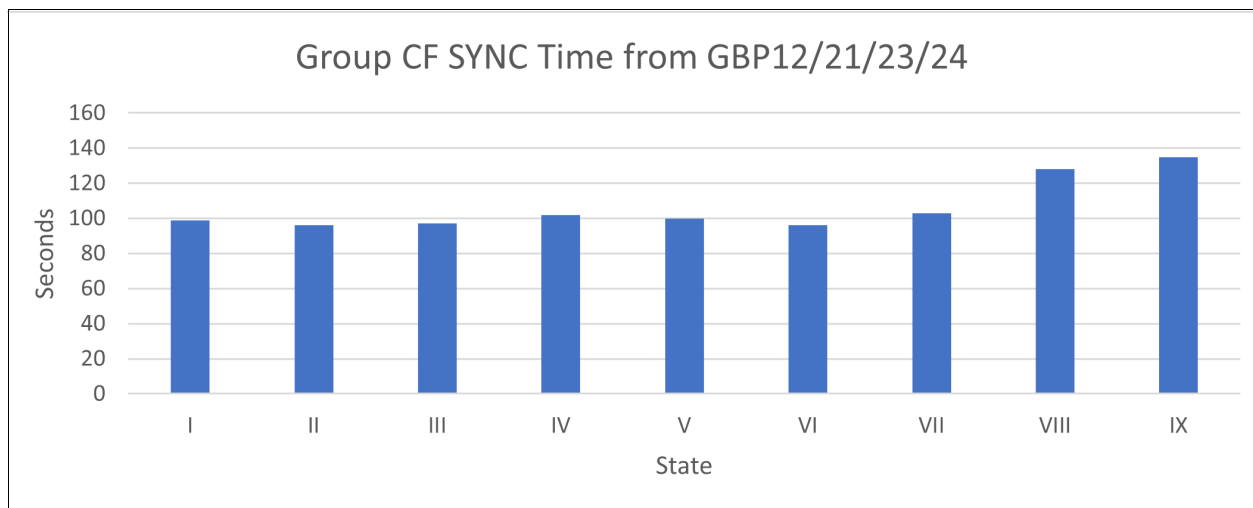


Figure 3-13 Group CF SYNC time from GBP12/21/23/24

It is clear from Figure 3-13 that state IX has the highest group CF SYNC time, 135 CPU seconds, which is a 36% increase compared to state I, which has 99 CPU seconds CF SYNC time.

The root cause of why the group ITR of state IX is not the best one is the higher than average number of CF requests during the measurement interval, which leads to higher CF utilization and results in longer CF SYNC service time. Figure 3-13 shows more CF SYNC CPU time in state IX than the rest of the states, so the group ITR of state IX is not the best one, although it is close.

### Summary of results

Drawing on the experiences from this exercise, we can summarize the following items:

- ▶ Small GBP residency times can mean frequent reclaims such that either directory or data entries do not stay long in the GBP, which might cause more synchronous I/Os because data is less likely to be found in the GBP when it is needed. The reclaim process itself also incurs a CPU cost, which is what happens in state I in this evaluation, which results in the lowest group ITR among the nine states.
- ▶ Large GBP residency times can mean infrequent reclaims on either directory or data entries such that they stay in the GBP for a long time, and the chance that the required data is found in the GBP is higher, and the workload performs fewer synchronous I/Os. The GBP is on the verge of zero GBP residency time, as shown in state VI or VIII in this exercise.
- ▶ Zero GBP residency time is expected to be the best performer because both directory and data entries stay in GBP forever. There are no reclaims at all, and there are the fewest synchronous I/Os.

Overall, the group ITR distribution in this exercise is 15,200 - 15,700 commits per second, which is a mere 3.3% difference and barely beats the measurement noise level. This situation occurred because reclaim processing was relatively small in the workload that we evaluated. Continue to monitor GBP shortage messages (DSNB325A or DSNB319A) in addition to the GBP residency time when tuning the GBP size.

## 3.5 System Recovery Boost: IBM z15 and later

IBM introduced the IBM zSystems Recovery Boost feature with IBM z15. It provides more capacity for clients to work more quickly through a backlog of work that accumulated while the system was down, which minimizes the impact of stopping and restarting the system.

Several types of boost are delivered with the IBM z15:

**Central processor (CP) speed boost**      Running subcapacity CPs at full-capacity speed during boost periods in boosting images only.

**Systems Integrated Information Processor (zIIP) boost**  
Making general-purpose work available to run on zIIP processors during the boost periods in boosting images only. Our evaluation focuses on this type only.

**Priced temporary boost capacity**      IBM GDPS® performance and parallelism enhancements.

Our evaluation of SRB focuses on the zIIP boost only. The duration of a zIIP boost is 1 hour.

SRB stage 3 is delivered with IBM z16 (driver 51). It enhances SRB in the following areas:

- ▶ **Middleware Region Startup Boost**

SRB can potentially be activated for 5 minutes to accelerate the restart of any started task. However, there is a limit of 30 minutes of Recovery Boost time per system per day. The Workload Manager (WLM) classification rules were enhanced to let you select those started tasks whose startup triggers the activation of SRB for that system. Because our tests were conducted on z15, we did not test this feature, but the inner workings are the same, but that the time and scope of what is boosted is different from the tests that are described.

- ▶ IBM HyperSwap® configuration load boost, which is not a focus in our evaluation.
- ▶ SVC Dump boost, which is not a focus in our evaluation.

In addition, SRB usability is also improved, which allows enablement and disablement of the SRB feature on a system, monitoring the status of SRB, and so on.

For more information about SRB, see *Introducing IBM Z System Recovery Boost*, REDP-5563.

### 3.5.1 Requirements

To leverage SRB, you must be using the following minimum levels of hardware and software:

- ▶ IBM z15 or later hardware
- ▶ Db2 12 for z/OS at FL 100 or later

### 3.5.2 Performance measurement

A number of in-house performance measurements were conducted to evaluate SRB.

## Measurement environment

The system that was used for the evaluation was an IBM z15 LPAR running z/OS 2.3. This LPAR has normally two CPs and two zIIPs that are assigned (without boost capability). An extra four zIIPs are reserved during the SRB period for a zIIP boost. Therefore, two CPs and six zIIPs are actively working on this LPAR during the SRB period, which lasts for 1 hour from IPL. In addition, the **B00ST** keyword in IEASYSxx can be set to the following values:

- ▶ B00ST=SYSTEM (default): SRB lasts 1 hour after IPL.
- ▶ B00ST=NONE: There is no SRB capability after IPL.

## Measurement scenario description

The following three scenarios were tested to evaluate the performance impact of SRB:

- ▶ Db2 restart after an unplanned LPAR outage
- ▶ Db2 restart after a planned LPAR outage
- ▶ Eliminating the workload backlog with and without SRB

## Results

The following sections describe the results for each of the scenarios.

### ***Scenario: Db2 restart after an unplanned LPAR outage***

Table 3-7 shows the performance impact of SRB for Db2 restarts after an unplanned LPAR outage.

*Table 3-7 Benefit of SRB to the Db2 restart after unplanned LPAR outage*

Event	No boost 2 CPs and 2 zIIPs (in seconds)	With boost 2 CPs and 2 zIIPs with 4 reserved zIIPs (in seconds)	Delta
IPL	168.66	165.38	-3.28
Db2 restart	258.28	149.15	-109.13
Log initialization	3.70	3.63	-0.07
Current status rebuild	9.01	8.50	-0.51
Forward log recovery	244.34	135.83	-108.51
Backward log recovery	1.24	1.19	-0.05

The timings for the different events in Table 3-7 are as follows:

- ▶ IPL: From “beginning” to “TSO Login”
- ▶ Db2 restart time is the sum of the following times:
  - Log initialization: From “-STA Db2” to message DSNJ099I
  - Current status rebuild: From message DSNJ099I to message DSNR004I
  - Forward log recovery: From message DSNR004I to message DSNR005I
  - Backward log recovery: From message DSNR005I to message DSN9022I

The impact of SRB on the Db2 forward log recovery phase during Db2 restart is significant in this scenario. SRB shortens the elapsed time of this phase. As a result, it speeds up the Db2 restart after an unplanned LPAR outage.

Figure 3-14 shows the general-purpose central processor (GPC) and zIIP CPU usage per minute for this scenario with and without SRB active.

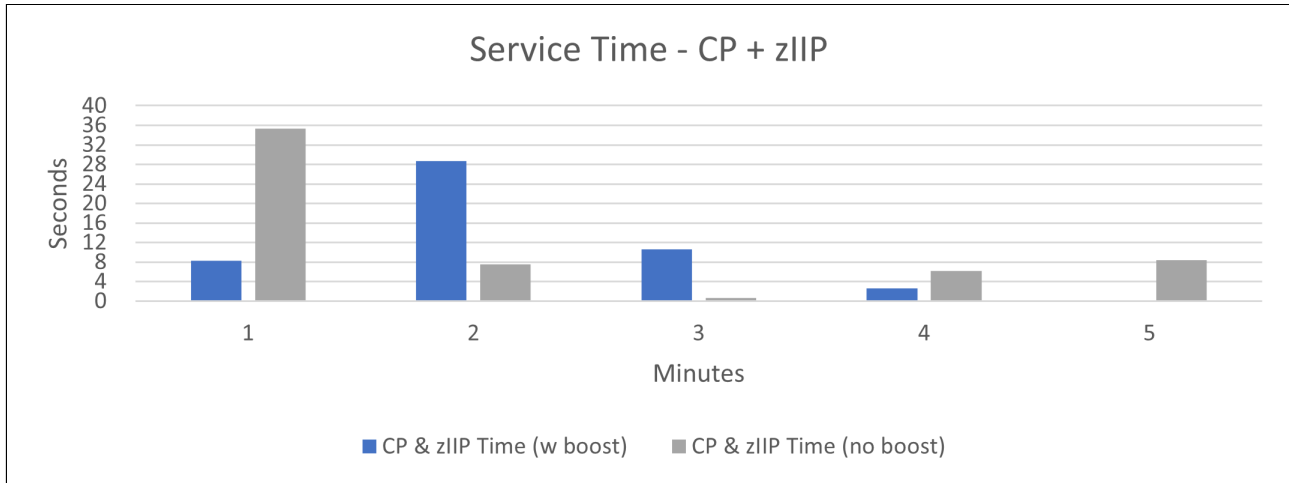


Figure 3-14 WLM service time for Db2 restart after unplanned LPAR outage

The forward log recovery phase of the Db2 restart process is designed to have fast log apply tasks working in parallel. Because there are more available zIIPs when SRB is used, which allow more tasks to be working concurrently, the results are an improved forward log recovery phase during Db2 restart.

**Scenario: Db2 restart after a planned LPAR outage**

Table 3-8 shows the performance impact of SRB on Db2 restart after a planned outage.

Table 3-8 Benefit of SRB for a Db2 restart after a planned LPAR outage

Event	No boost 2 CPs and 2 zIIPs (in seconds)	With boost 2 CPs and 2 zIIPs with 4 reserved zIIPs (in seconds)	Delta
IPL	161.84	162.00	+0.16
Db2 restart	7.16	5.02	-2.14
Log initialization	3.20	2.71	-0.49
Current status rebuild	2.11	2.10	-0.01
Forward log recovery	0.01	0.00	-0.01
Backward log recovery	1.84	0.21	-1.63

The timings for the events that are listed in Table 3-8 are as follows:

- ▶ IPL: From “beginning” to “TSO Login”
- ▶ Db2 restart is the sum of the following times:
  - Log initialization: From “-STA Db2” to message DSNJ099I
  - Current status rebuild: From message DSNJ099I to message DSNR004I
  - Forward log recovery: From message DSNR004I to message DSNR005I
  - Backward log recovery: From message DSNR005I to message DSN9022I

Because the forward recovery phase is expected to be short after a planned LPAR (and Db2 shutdown), the impact of SRB is minimal in this scenario.

**Scenario: Eliminating the workload backlog with and without SRB**

SRB provides more capacity for working quickly through a backlog of work that accumulated while the system was down. Reducing the workload backlog leads to a shorter business impact.

Figure 3-15 illustrates the workload external throughput rate (ETR) over time for a scenario without SRB, where approximately 10 minutes after the system undergoes IPL, Db2 is restarted to work on eliminating the backlog that is accumulated while the system was down. It is evident that the ETR gradually drops as time goes on, meaning less and less backlog is cleared as time passes.

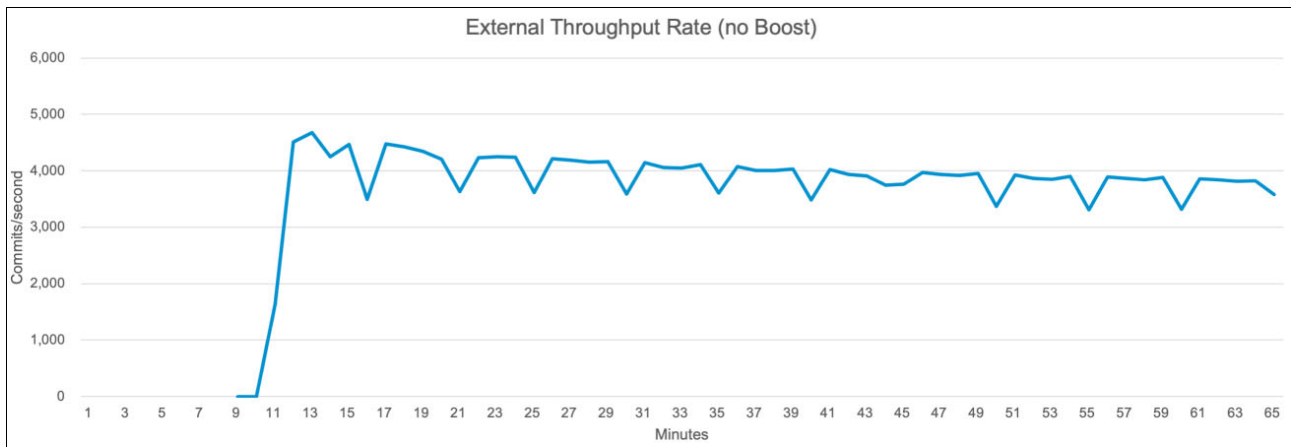


Figure 3-15 Throughput rate when eliminating the backlog without SRB

Concurrently, the transaction elapsed time is getting longer, as shown in Figure 3-16.

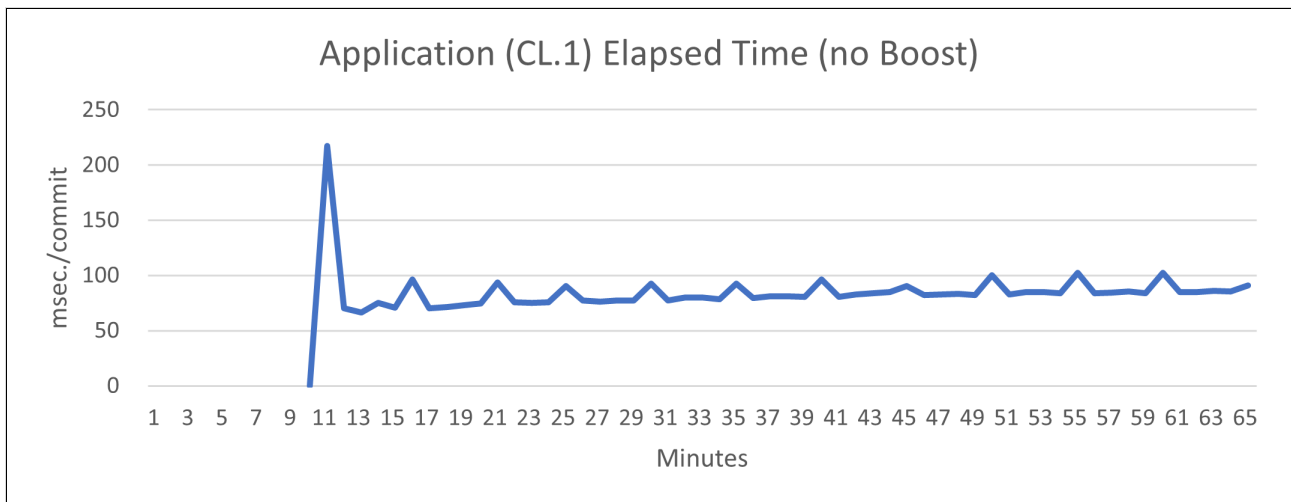


Figure 3-16 Application elapsed time when eliminating the backlog without SRB

Dropping ETR and elongated transaction elapsed time is the result of nearly saturated CP and zIIP utilization, as shown in Figure 3-17 and Figure 3-18 when the system undergoes IPL without SRB capability. Only two CPs and two zIIPs are available to process all the work.

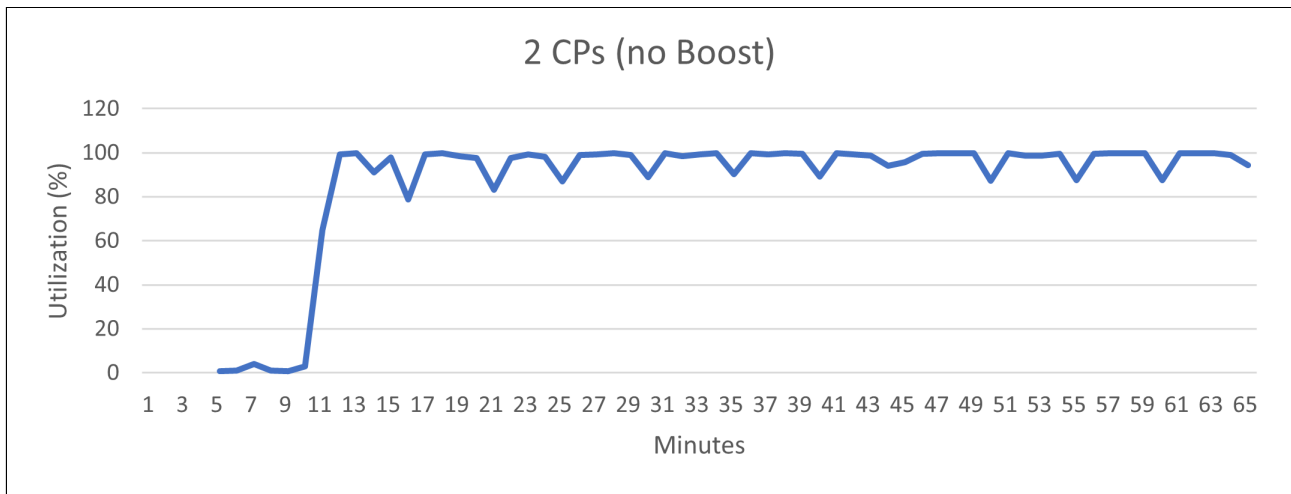


Figure 3-17 General processor utilization when eliminating the backlog without SRB

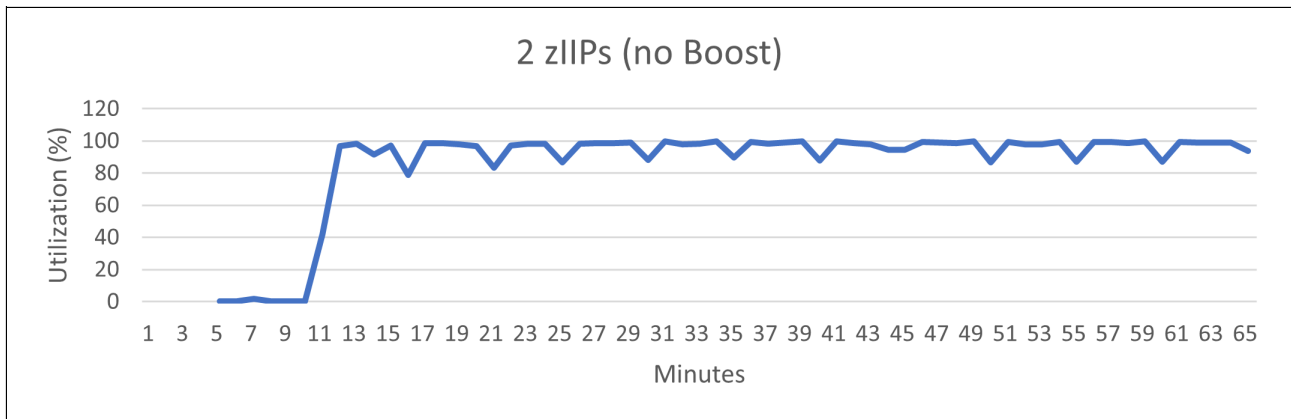


Figure 3-18 zIIP processor utilization when eliminating the backlog without SRB

When the backlog is being cleared in a CPU-bound environment, the workload is starved for processing power.

Now, look at a system that undergoes IPL with SRB capability. With SRB, two CPs and six zIIPs are working for 1 hour after IPL (the boost period) to clear the backlog.

Almost immediately, a reduction in various Db2 internal latch suspension times is observed. Db2 page latch suspension time is also reduced, as shown in Figure 3-19 on page 65.



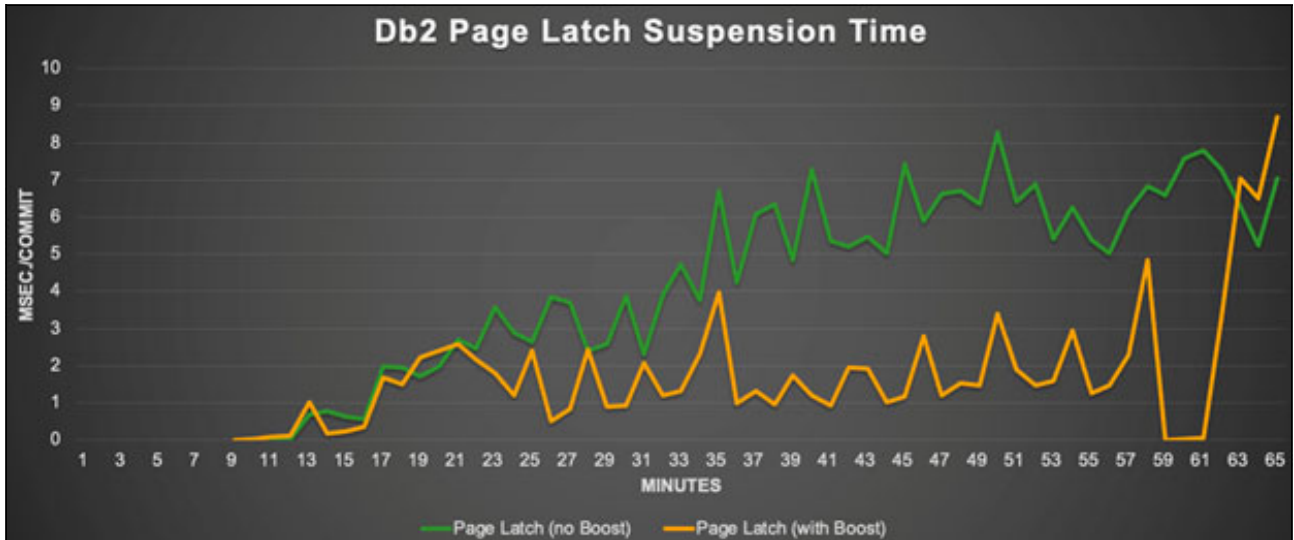


Figure 3-19 Db2 page latch wait when eliminating the backlog with and without SRB

Reducing both Db2 internal latch and page latch suspension time leads to more stability of the transaction elapsed time, as shown in Figure 3-20.

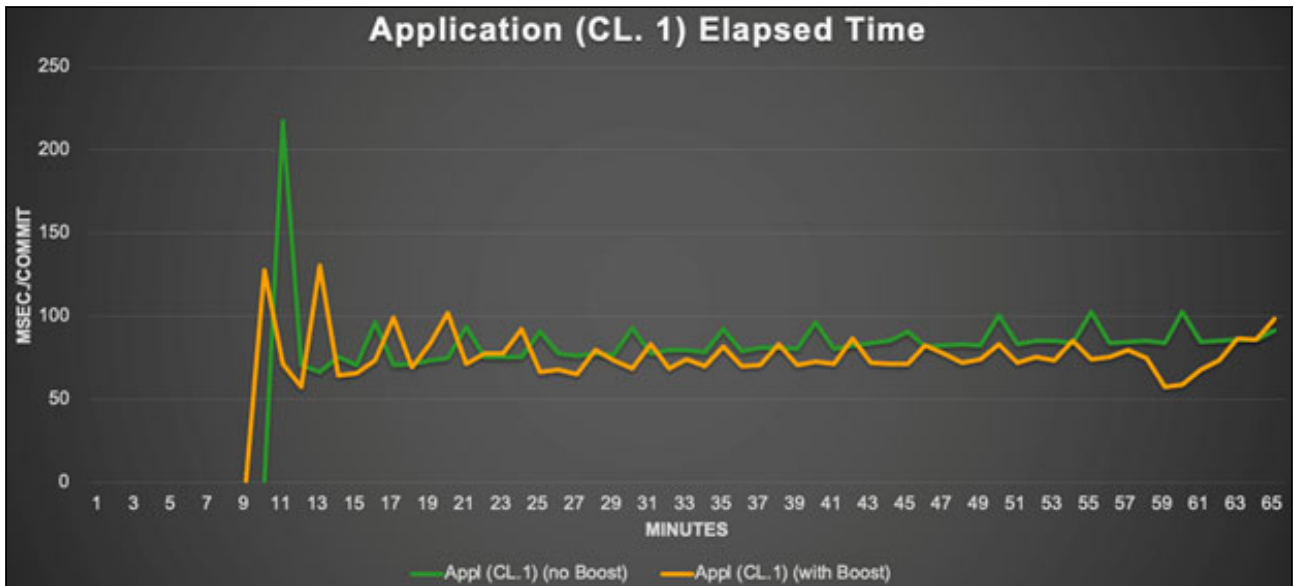


Figure 3-20 Application elapsed time when eliminating the backlog with and without SRB

The shortened Db2 application elapsed time drives up ETR with the same number of user threads and user think time driving the workload. As a result, the backlog is eliminated sooner and faster with SRB than without it, as shown in Figure 3-21.

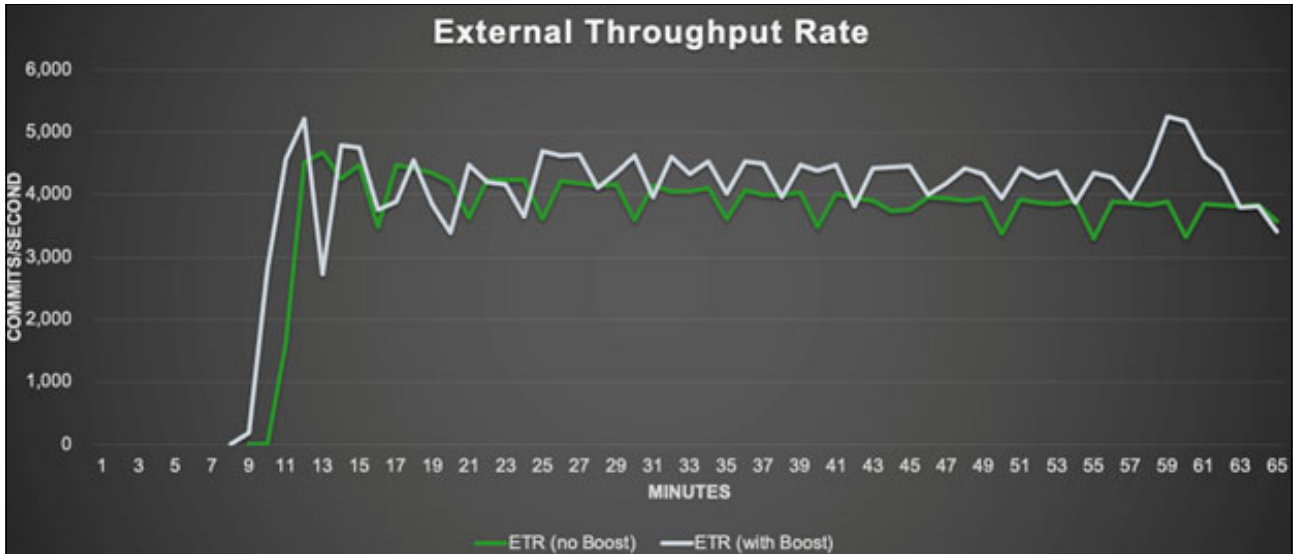


Figure 3-21 External throughput when eliminating the backlog with and without SRB

After 1 hour, SRB is no longer active, and the system goes back to a two CPs and two zIIPs configuration. Thereafter, the Db2 application elapsed time and workload ETR exhibit the same behavior.

Figure 3-22 shows that the LPAR (CP and zIIP) MVS is busy from the time the system undergoes IPL until 65 minutes after IPL.

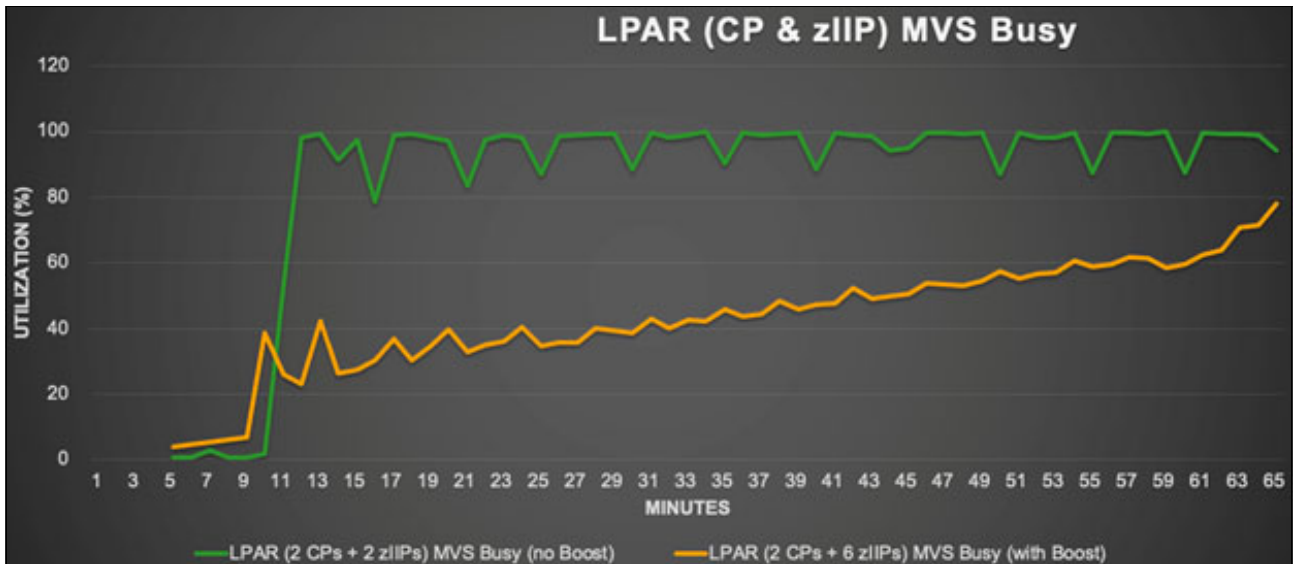


Figure 3-22 MVS busy when eliminating the backlog with and without SRB

The LPAR (CP and zIIP) is less stressed because it is relieved by SRB, which provides extra zIIPs working together through the boost period to clear the backlog.

## Summary of results

Based on the results of the three scenarios, we offer the following key observations about the performance impact of SRB:

- ▶ For the “Db2 restart after unplanned LPAR outage” scenario, the forward log recovery phase of Db2 restart is designed to have tasks working together in parallel. The increased number of available zIIPs that come with SRB allow more tasks to be working simultaneously, which results in a much improved forward log recovery phase during Db2 restart.
- ▶ For the “Db2 restart after planned LPAR outage” scenario, SRB has only a minimal performance impact in this scenario because Db2 restart in this scenario is mostly I/O bound.
- ▶ For the “Eliminating the workload backlog with and without SRB” scenario, without SRB, the Db2 workload backlog is cleared in a CPU-bound environment. As a result, the backlog grinds through in a strained and inefficient fashion. With SRB, clearing this backlog is no longer CPU-bound. CPU-bound processes, such as thread allocation and local buffer warm up, benefit from SRB. SRB relieved Db2 internal latch contention and reduced Db2 page latch suspension time, which resulted in improved Db2 transaction elapsed time. As a result, the workload achieves a higher transaction throughput rate throughout the boost period, and the backlog is eliminated sooner, faster, and more efficiently than on a system without SRB.

### 3.5.3 Setup and monitoring information

This section provides some general information for enabling SRB and monitoring its usage.

#### How to set up System Recovery Boost

To set up SRB, go to the Hardware Management Console (HMC), as shown in Figure 3-23.

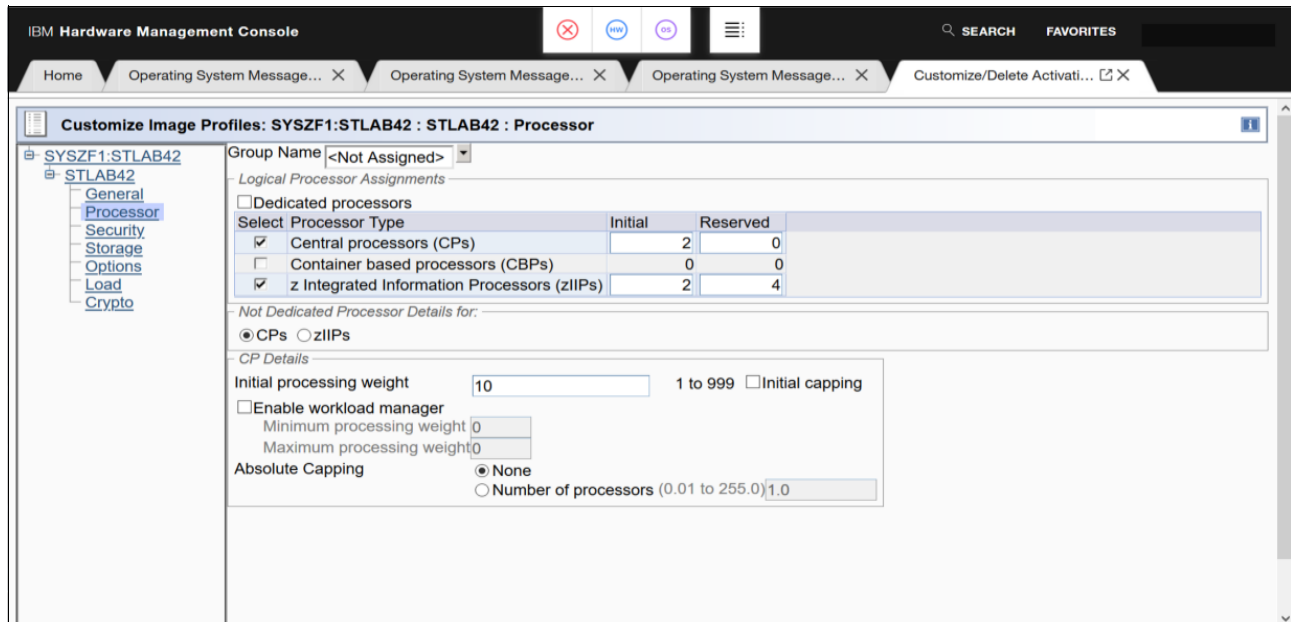


Figure 3-23 How to setup System Recovery Boost

Complete the following steps:

1. Clear **Dedicated processors**.
2. Enter a nonzero value in the z Integrated Information Processors (zIIPs) Reserved field.
3. Edit IEASYSxx and ensure that the **BOOST** keyword is set to SYSTEM, which is the default value. BOOST=SYSTEM causes SRB to last until 1 hour after IPL.

The following messages indicate that SRB is in effect:

```
IEA672I IPL speed boost is not activated – processor is already full speed
IEA675I IPL zIIP boost is active with 4 transient zIIP cores
IWM064I BOOST ACTIVATED
D IPLINFO,BOOST > IEE255I SYSTEM PARAMETER ‘BOOST’: SYSTEM
```

1 hour later, the following messages indicate that the SRB duration ended:

```
IEA678I All IPL boosts have ended
IWM063I WLM POLICY WAS REFRESHED
IWM064I BOOST ENDED
```

### How to monitor System Recovery Boost

An easy way to identify whether SRB was active at a certain time is to look at the RMF Workload Activity Report. The Service Policy page has a “BOOST” heading that indicates whether SRB was inactive or active at the end of the interval during IPL, shutdown, or recovery. If SRB is active, one of the following boost values is shown to indicate the type of boost that is active:

I	zIIP boost.
S	Speed boost.
A	zIIP and speed boost were both active.

## 3.5.4 Usage considerations

A zIIP boost is the focus of our SRB evaluation. It provides extra processing capacity to speed up a Db2 restart after an unplanned LPAR outage and to work more quickly through a backlog of work that accumulated while the system was down. Therefore, a zIIP boost of SRB is most effective when tasks or backlog work are CPU-bound by nature. When tasks or work are not constrained by processing power, such as when they are I/O-bound, a zIIP boost might not be effective.

APAR PH34550 addresses a Db2 RLF issue with SRB. Make sure that the PTF for this APAR is applied when Db2 is restarted across the SRB period.

## 3.5.5 Conclusion

The focus of this study is on exploring the effects of the zIIP boost option of the SRB feature. It provides extra processing power by allowing zIIP processors to work like general processors during a 1-hour SRB period after a system IPL. Therefore, the extra processing capability helps clients to work more quickly through a backlog of work that accumulated while the system was down.

When Db2 was restarted after a planned LPAR outage, SRB had minimal performance impact because the Db2 restart is mostly I/O-bound in this case, and adding more CPU power does not speed up the restart time.

When Db2 was restarted after an unplanned LPAR outage, we observed faster Db2 restart time while the IPL time remained unchanged. The forward log recovery phase of Db2 restart is designed to have tasks working together in parallel so that it can leverage more available zIIPs during SRB boost time to allow more tasks to work concurrently.

## 3.6 REORG table space with variable length records on IBM z15 with IBM Z Sort

IBM z15 provides a new on-chip sort accelerator that is known as the IBM Integrated Accelerator for IBM Z Sort (Z Sort). IBM z15 also provides the new **SORT LISTS (SORTL)** instruction, which allows DFSORT to take the hardware-accelerated approach to sort, which reduces the CPU cost and shortens the elapsed time. The Db2 **REORG** utility implements the interface to leverage the IBM z15 features when DFSORT is invoked to reorganize a Db2 table space with variable length records (VLRs).

### 3.6.1 Requirements

To leverage this enhancement, you need the following hardware and software:

- ▶ IBM z15 or later
- ▶ z/OS 2.3 or later
- ▶ Db2 12 for z/OS at FL 500 or later
- ▶ Db2 12 APAR PH28183
- ▶ DFSORT APAR PH03207

### 3.6.2 Performance measurement

As with other important Db2 enhancements, a performance study was conducted to evaluate this feature.

#### Measurement environment

The performance evaluation was done on both IBM z14 and IBM z15 systems. Both systems were set up with z/OS 2.3, Db2 12, four general-purpose processors, one zIIP engine, and 512 GB of memory.

#### Measurement scenario description

The table that is used for this test uses a partition-by-range (PBR) table space with 20 partitions and the **COMPRESS YES** option. The table has 15 VARCHAR columns, and the average record length is 243 bytes. Two indexes are defined on the table: One index is a partitioned index (PI), and the other is a non-partitioned index (NPI). The table is loaded with 100 million rows in random order against the clustering index before running the **REORG**.

Then, the **REORG TABLESPACE** utility is run with the **SORTDATA**, **SHRLEVEL CHANGE**, and **KEEPDICTIONARY** options. Two sorts are performed during the utility run time: sorting data VLR, and sorting when building the indexes (fixed-length record (FLR)). Both sorts use DFSORT, but only the VLR part is Z Sort eligible.

The IBM z15 LPAR is equipped with enough memory so that Z Sort can contain the sort entirely in memory in this test.

Two **REORG** performance comparisons are done in this test:

- ▶ Scenario 1 compares the IBM z14 measurement with the IBM z15 measurement with Integrated Accelerator for IBM Z Sort. This scenario shows how much performance improvement there is when moving from IBM z14 to IBM z15 with Z Sort.
- ▶ Scenario 2 compares an IBM z15 measurement with Z Sort disabled to an IBM z15 measurement with Z Sort enabled. This scenario shows how much performance improvement might be achieved by using Z Sort itself on IBM z15.

For both scenarios, measurements are repeated: the first run uses a short clustering key length (33 bytes), and the second run uses a longer clustering key length (165 bytes).

The performance numbers that are used for comparison are ELAPSE, SORTCPU, and STEPCPU from the System Management Facility (SMF) 16 and SMF 30 records, and above-the-bar (ATB) real memory usage from the job outputs.

## Results

The measurement results can be summarized as follows:

- ▶ Scenario 1 (IBM z15 with Z Sort versus IBM z14)
  - Short clustering key length

For the VLR part, the improvement is up to 17% in ELAPSE and 40% in SORTCPU time. For the overall **REORG** job, the improvement is up to 11% in ELAPSE and 17% in STEPCPU time. z15 with Z Sort uses 91% more ATB real memory.
  - Longer clustering key length

For the VLR part, the improvement is up to 23% in ELAPSE and 34% in SORTCPU time. For the overall **REORG** job, the improvement is up to 15% in ELAPSE and 20% in STEPCPU time. z15 with Z Sort uses 66% more ATB real memory.
- ▶ Scenario 2 (IBM z15 with Z Sort versus IBM z15 no Z Sort)
  - Short clustering key length

For the VLR part, the improvement is up to 7% in ELAPSE and 32% in SORTCPU time. For the overall **REORG** job, the improvement is up to 4% in ELAPSE and 9% in STEPCPU time. IBM z15 with Z Sort uses 91% more ATB real memory.
  - Longer clustering key length

For the VLR part, the improvement is up to 3% in ELAPSE and 25% in SORTCPU time. For the overall **REORG** job, the improvement is up to 1% in ELAPSE and 8% in STEPCPU time. IBM z15 with Z Sort uses 66% more ATB real memory.

### 3.6.3 Usage considerations

DFSORT with Z Sort is shipped as disabled by default. You can enable it through a DFSORT installation option or you can modify your JCL to enable it for that job. In this test, we specify **OPTION ZSORT** in the **DFSPARM DD** statement in the JCL to enable it.

Db2 also has the **UTILS\_USE\_ZSORT=YES/NO** subsystem parameter (NO is the default) to control the usage of Z Sort for the **REORG TABLESPACE** utility. When **UTILS\_USE\_ZSORT=YES**, if the Db2 **REORG TABLESPACE** utility detects at run time that Z Sort is enabled and the architecture supports Z Sort from DFSORT, then Db2 uses Z Sort.

DFSORT requires the use of 64-bit memory objects to use Z Sort.

Z Sort tends to use more ATB memory. Whether the memory is large enough to contain the sort affects Z Sort performance.

The following two messages in the job outputs are helpful for determining z Sort status:

- ▶ ICE267I indicates whether Z Sort is used.
- ▶ ICE399I indicates how much memory is used.

### 3.6.4 Conclusion

When you are using IBM z15 and later, Z Sort for DFSORT is a good feature to enable and test. Currently, only the Db2 **REORG TABLESPACE** utility with VLR uses it, but Db2 might expand the usage to **REORG TABLESPACE** with FLR or other utilities, such as **LOAD**, in the future.

## 3.7 Leveraging Z Sort by SQL sort

In addition to the **REORG** utility, Db2 SQL sort operations can also leverage IBM Integrated Accelerator for Z Sort. Z Sort uses the **SORTL** instruction, which is an IBM z15 problem state instruction that can turn up to 128 lists of unsorted input data into one or more lists of sorted output data. It also provides a means to merge multiple lists of sorted-input data into a single list of sorted-output data. Db2 added support for **SORTL** for some **ORDER BY** and **GROUP BY** sorts.

Besides the general requirements for any SQL sort to use **SORTL**, not all sorts benefit from using **SORTL**. An individual sort operation can leverage **SORTL** when the following conditions are met:

- ▶ It must be part of a Db2 12 plan or package.
- ▶ The sort key size is  $\leq$  136 bytes, and the data size is  $\leq$  256 bytes. These sizes are not “hard rule” numbers. They can be adjusted to be larger depending on the setting of the **SRTPOOL** subsystem parameter and when the number of sort records is known.
- ▶ The sort key size is equal to the data size (all the data columns are key columns with no variable length fields).

### 3.7.1 Requirements

For any SQL sort to be considered for Z Sort usage, the system must meet the following requirements:

- ▶ IBM z15 and later
- ▶ Db2 12 at FL 100 or later
- ▶ Db2 APARs PH31684 and PH37239

### 3.7.2 Performance measurement

This section describes the environment in which we conducted our evaluation, the scenarios that we used, and the results.

## Measurement environment

The following system setup was used:

- ▶ IBM z15 hardware with three GPCs and three zIIPs
- ▶ Direct access storage device (DASD): DS8800 with 512 GB of storage
- ▶ z/OS 2.4
- ▶ Db2 12 FL 500

## Measurement scenario description

For this evaluation, we use individual **SELECT** queries with an **ORDER BY** clause with various sort key lengths and **SRTPOOL** sizes.

## Results

The performance measurements showed an average 8% (0 - 30%) class 2 CPU time reduction and average 0.3% class 2 elapsed time reduction with qualified sort-intensive queries. The measurements show that the percentage of the performance improvement is affected by the following factors, which are ordered by the significance of their impact on the query performance:

1. The sort pool size, which is determined by the **SRTPOOL** subsystem parameter
2. The number of rows sorted
3. The sort key size

Generally, for a qualified query with the same number of rows sorted and the same sort key size and sort data size, the larger the sort pool size, the greater the reduction in CPU and elapsed time. With the default **SRTPOOL** setting (10000KB) in Db2 12, the more rows sorted, the more performance savings. The smaller the sort key size is, the more performance savings. However, with an **SRTPOOL** size of 128000, the impact of the number of rows and sort key size is not as obvious as with smaller **SRTPOOL** sizes.

Table 3-9 and Table 3-10 on page 73 provide more details about the performance improvements when **SORTL** is used for different **SRTPOOL** sizes, which varies the number of rows that are sorted, and the sort key length.

Table 3-9 Db2 class 2 CPU time improvement (%) for SRTPOOL size 10 M

Number of rows sorted	Sort key size less than 40 bytes	Sort key size more than 40 bytes
10	Up to 13%	Up to 8%
100	Up to 8%	Up to 7%
1000	Up to 17%	Up to 12%
10000	Up to 21%	Up to 15%
100000	Up to 35%	Up to 29%
1000000	Up to 32%	Up to -29%
10000000	Up to 35%	Up to 30%



Table 3-10 Db2 class 2 CPU time improvement(%) for SRTPOOL size 128 M

Number of rows sorted	Sort key size less than 40 bytes	Sort key size more than 40 bytes
10	Up to 30%	Up to 24%
100	Up to 24%	Up to 20%
1000	Up to 20%	Up to 14%
10000	Up to 20%	Up to 13%
100000	Up to 19%	Up to 6%
1000000	Up to 33%	Up to 16%
10000000	Up to 33%	Up to 21%

### 3.7.3 Monitoring information

Both the Db2 accounting (IFCID 3) and statistics (IFCID 2) trace records added instrumentation to show the number of SQL sort operations that are performed and how many of them use **SORTL**, which means that the information is available at the transaction and system level. Figure 3-24 shows an excerpt of an IBM OMEGAMON for Db2 Performance Expert (OMPE) Db2 accounting report with the new counters:

- ▶ RDS SORT PERFORMED (QXSTSRT)
- ▶ RDS SORTL USED (QXSTSRTL)

DYNAMIC SQL STMT	AVERAGE	TOTAL
REOPTIMIZATION	0.00	0
NOT FOUND IN CACHE	1.00	1296
FOUND IN CACHE	7.50	9720
IMPLICIT PREPARES	0.00	0
PREPARES AVOIDED	0.00	0
CACHE_LIMIT_EXCEEDED	0.00	0
PREP_STMT_PURGED	0.00	0
STABILIZED PREPARE	0.00	0
CSWL - STMTS PARSED	0.00	0
CSWL - LITS REPLACED	0.00	0
CSWL - MATCHES FOUND	0.00	0
CSWL - DUPLS CREATED	0.00	0
RDS SORT PERFORMED	8.50	11016
RDS SORTL USED	6.38	8262
ZAI STBLZD PREPARE	0.00	0
ZAI SORT FB USED	0.00	0

Figure 3-24 OMPE Db2 accounting report for SORTL usage

Figure 3-25 shows the section of the OMPE statistics report where this information can be found.

MISCELLANEOUS	VALUE
HIGH LOG RBA	0000000000189AD2A784D
BYPASS COL	0.00
MAX SQL CASCADING LEVEL	0.00
MAX STOR LOB VALUES (MB)	0.00
MAX STOR XML VALUES (MB)	0.00
ARRAY EXPANSIONS	0.00
SPARSE IX DISABLED	0.00
SPARSE IX BUILT WF	0.00
NO DM CALL RIDL/LPF	0.00
FETCH 1 BLOCK ONLY	2160.00
RDS SORT PERFORMED	11016.00
RDS SORTL USED	8262.00
ZAI STABILIZED PREPARE	0.00
ZAI SORT FEEDBACK USED	0.00
FTB THRESHOLD	0.00
FTB CRITERIA MEET	0.00
FTB TRAVERSE ABOVE THE THRESHOLD	0.00
FTB TOTAL MEMORY ALLOCATION	0.00
FTB IN THE PREVIOUS OPTIMIZATION	0.00
FTB IN THE CURRENT OPTIMIZATION	0.00

Figure 3-25 OMPE Db2 accounting statistics report for SORTL usage

### 3.7.4 Usage considerations

The performance numbers show that using **SORTL** can help to reduce the CPU cost and improve the elapsed time for eligible workloads. Even though **SORTL** does not exceed the maximum amount of storage that it is allowed to use, which is indicated by the **SRTPOOL** subsystem parameter, **SORTL** typically benefits from using more storage. So, threads that perform SQL sort operations that use **SORTL** tend to use more storage to achieve the CPU and elapsed time improvements that we described in the previous sections.

However, when this storage is freed, threads that use **SORTL** can incur extra *ssnmMSTR* SRB or *ssnmDIST* SRB system CPU time from thread deallocation on systems that specify the default setting of AUTO (or ON) for the **REALSTORAGE\_MANAGEMENT** subsystem parameter. Because the storage discard processing happens only at thread deallocation time, this situation can be alleviated by optimal thread reuse. In addition, ATB storage management was enhanced in Db2 13, which eliminates the need for the **REALSTORAGE\_MANAGEMENT** subsystem parameter.

**SORTL** usage can help improve performance for qualified queries. One of the primary requirements for using Z Sort is providing enough virtual, real, and auxiliary storage. To provide the most benefit from **SORTL** usage, the default **SRTPOOL** subsystem parameter setting for Db2 13 is changed from 10000KB to 20000KB.

## 3.8 Large object compression

Db2 13 continues to support large object (LOB) compression with usability enhancements for **DSN1COMP**. The **DSN1COMP** stand-alone utility estimates space savings that can be achieved by data compression in table spaces, including LOB table spaces and indexes.

LOB compression in Db2 is not dictionary-based like table space compression. On IBM zEC12, z13, and z14, the IBM zEnterprise Data Compression (zEDC) express card is required to perform LOB compression. IBM z15 and later use the Integrated Accelerator for zEDC to perform compression. The Integrated Accelerator for zEDC replaces the existing zEDC Express adapter in the I/O drawer. IBM z15 uses one accelerator chip per processor, and it is shared by all processor cores. This new design offers low latency and high bandwidth for compression. The primary goal of compression is to save disk space. However, there is an overhead for compressing the LOB data during insert operations and an overhead for decompressing LOB data during select operations.

Good candidates for LOB compression are LOB columns that store data in formats such as DOCX, TXT, XML, and HTML. For binary files with a low compression ratio, Db2 recognizes that no benefit is gained from compression, and it does not compress the data.

You can use the **DSN1COMP** stand-alone utility to estimate the compression ratio and the number of data pages that are saved before the compression of the LOBs is implemented.

### 3.8.1 Performance measurement

Several measurements were conducted to evaluate the Db2 LOB compression feature, with a focus on select and insert performance, by using CPU utilization and compression ratio as the key performance measurement indicators. Generated files with different compression ratios and common data type files such as TXT, XML, DOCX, JPG, PDF, PNG, and PPT were evaluated to understand the compression ratio for different data types that use the same Db2 level running on IBM z14 and IBM z15 hardware. On both IBM z14 and IBM z15 hardware, Db2 13 at FL 500 was used for the tests.

### IBM z15 offers a better compression ratio

Figure 3-26 presents a comparison of the compression ratio for a LOB table space with different data types on IBM z14 and IBM z15 hardware. The compression ratio that is reported by `DSN1COMP` indicates the space saving by compression. The data shows the Integrated Accelerator for zEDC provides a better compression ratio in all cases compared to the zEDC card on IBM z14. For example, for the `compress50.txt` case, the Integrated Accelerator for zEDC achieves a compression ratio of 49%, but the zEDC on IBM z14 can achieve only a 24% compression ratio. For some data types, such as TIF, PNG, or PDF, there is not much difference in the compression ratio on either hardware level. This fact also is true for the mixed case in which random data types are inserted into the LOB tables.

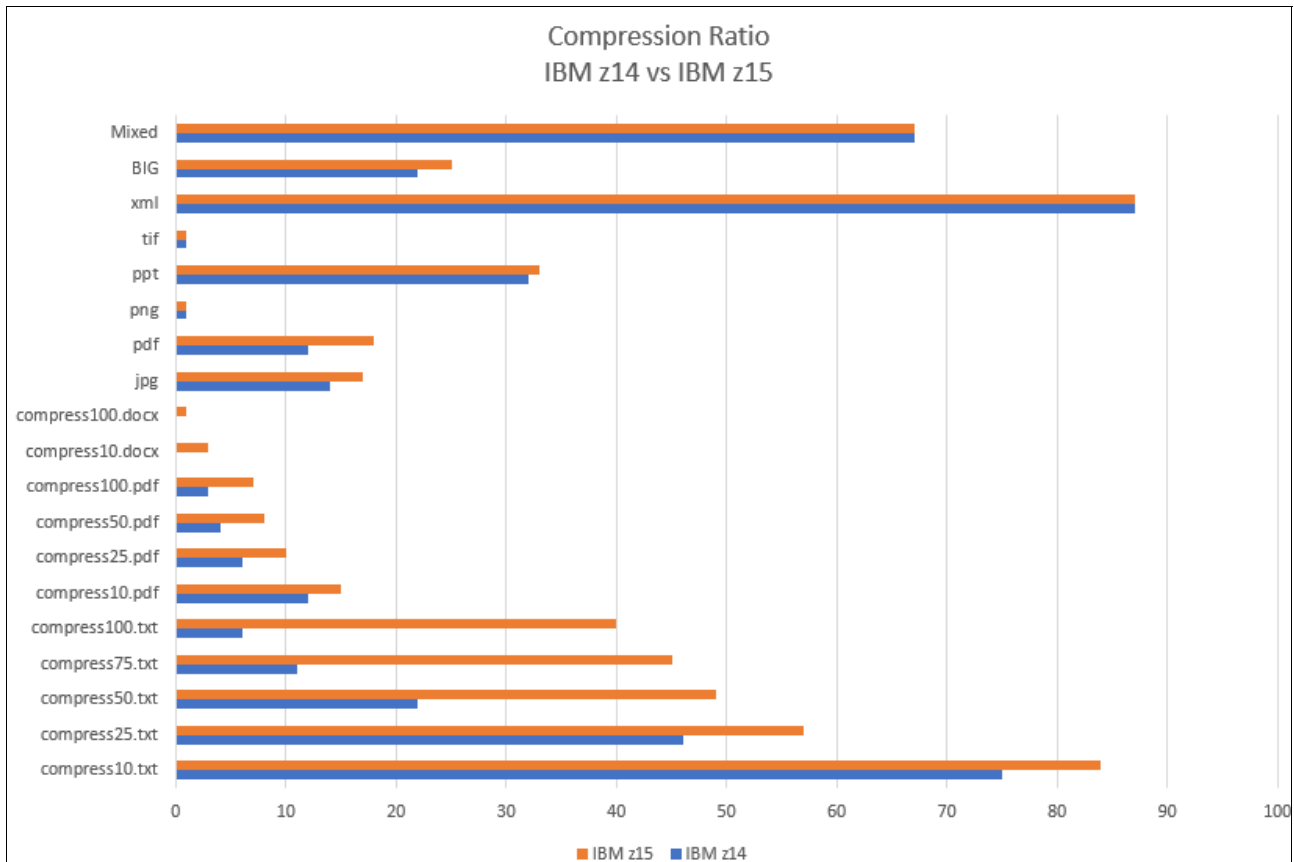


Figure 3-26 Compression ratio on IBM z14 and IBM z15

### IBM z15 offers faster compression and decompression time

Table 3-11 on page 77 presents the percentage improvement for insert and select operations with different compression ratio files on IBM z14 and IBM z15. The class 3 LOB compression suspension time is the time waiting for compression or decompression, and the main metric for comparison between IBM z14 and IBM z15. The data shows the Integrated Accelerator for zEDC on IBM z15 performs much better than the zEDC PCIe cards on IBM z14.

For insert operations, LOB compression works up to 70% better on IBM z15 for most common file types. With the mixed data files scenario, the insert operations perform 70% better on IBM z15 hardware than they do on IBM z14 with zEDC Express cards. For select operations, IBM z15 hardware offers 27% better performance in decompression time for mixed file type scenarios. With low compression ratio LOBs or TIF files, there is no improvement on decompression.

Table 3-11 Compression ratio and wait time improvement of insert or select on z15

File name or format	Compression ratio		Class 3 LOB compression suspension time (percent improvement)	
	IBM z14	IBM z15	Insert	Select
Compress10.txt	75	84	-72.87	-56.57
Compress25.txt	46	57	-66.83	-59.29
Compress50.txt	22	49	-66.02	-62.06
Compress75.txt	11	45	-67.95	-51.70
Compress100.txt	6	40	-64.40	-56.71
Compress10.pdf	12	15	-56.40	-63.33
Compress25.pdf	6	10	-59.00	-58.43
Compress50.pdf	4	8	-67.56	-59.49
Compress100.pdf	3	7	-7.06	-60.85
JPG	14	17	-57.08	-13.33
PPT	32	33	-60.57	-60.19
TIF	1	1	-70.88	0
XML	97	95	-77.35	-75.47
Large	22	25	-71.70	0
Mixed	67	67	-70.85	-26.79

Compression time reduction helps to improve LOB compression throughput on IBM z15. Figure 3-27 shows the results from the measurements of the workload with 10, 20, and 40 concurrent users that perform insert operations with the mixed data files scenario. It shows the trend that you can achieve better throughput with IBM z15 hardware.

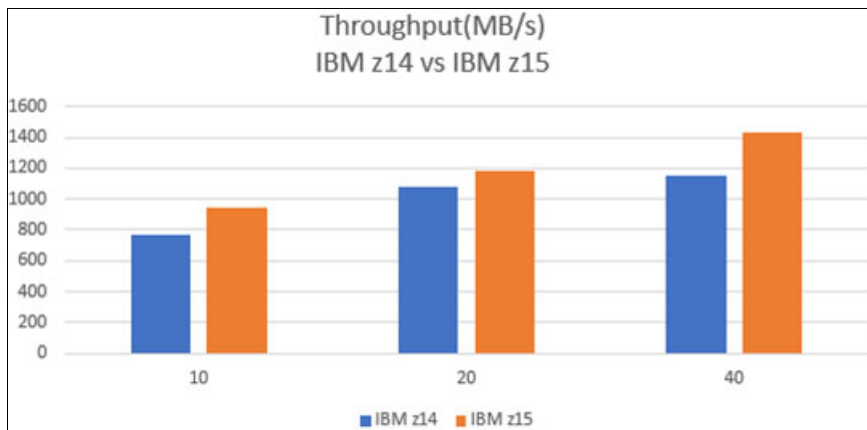


Figure 3-27 Throughput of LOB compression on IBM z14 and IBM z15

When 20 concurrent threads are used to perform insert operations, IBM z15 achieves a throughput of 1418 MBps, but IBM z14 achieves only 1143 MBps. Overall, the Integrated Accelerator on IBM z15 offers 21% - 39% more throughput in MBps than the zEDC on IBM z14.

## IBM z15 offers CPU cost reduction

The Integrated Accelerator for zEDC on IBM z15 hardware also helps to reduce the CPU cost of insert and select operations against tables that use LOB compression. Table 3-12 presents the percentage of class 2 CPU improvement of insert and select operations for LOB compression with different file types. The data shows LOB compression on IBM z15 uses 10 - 30% less CPU than it takes on IBM z14 across different file types. In the mixed scenario, the CPU time improvement of compressing LOBs during insert operations is 10%, and the CPU time improvement for select operations to retrieve and decompress them is 31%.

Table 3-12 Compression ratio and CPU improvement on IBM z15

File name	Compression ratio		CPU time (percent improvement)	
	IBM z14	IBM z15	Insert	Select
Compress10.txt	75	84	-15.95	-29.17
Compress25.txt	46	57	-18.7	-31.15
Compress50.txt	22	49	-21.36	-30.89
Compress75.txt	11	45	-21.75	-27.97
Compress100.txt	6	40	-22.17	-28.57
Compress10.pdf	12	15	-20.39	-30.65
Compress25.pdf	6	10	-17.59	-28.21
Compress50.pdf	4	8	-18.00	-30.77
Compress100.pdf	3	7	-13.40	-30.17
JPG	14	17	-25.58	-30.00
PPT	32	33	-16.57	-29.75
TIF	1	1	-10.08	-30.63
XML	97	95	-10.90	-31.03
Large	22	25	-5.05	-31.25
Mixed	67	67	-10.65	-31.40
PDF	19	19	-16.52	-13.98

## IBM z15 can handle larger compression work

RMF Monitor III can gather SMF record type 74 subtype 9 data that is required for the PCIe Activity report. You can use the RMF post-processor to produce a report about the PCIe activity. Figure 3-28 on page 79 presents a section of a PCIe report for the zEDC card on IBM z14. The data shows that the zEDC card is only 14% used. However, the compression throughput is smaller than the compression request rate, so some compression requests are queued before the compression was performed.

▼ Hardware Accelerator Activity					
Function ID	Time Busy %	Adapter Utilization	Work Units Processed Rate	Request Executi	
0033	13.0	14.2	142389	291	
0044	13.0	14.0	139660	291	

▼ Hardware Accelerator Compression Activity					
Function ID	Compression Request Rate	Compression Throughput	Compression Ratio		
0033	446	228	1.27		
0044	446	228	1.27		

Figure 3-28 PCIe activity report on IBM z14

Conversely, the INPUT/OUTPUT PROCESSORS section report in the RMF I/O QUEUING ACTIVITY report shows the activity for the Integrated Accelerator for zEDC. Figure 3-29 shows that on IBM z15, compression busy % is 12%, but there is no queue. Also, the compression chip is only 0.92% busy, so the Integrated Accelerator for zEDC can handle more compression work than the zEDC card on IBM z14.

- INITIATIVE QUEUE -			----- IOP UTILIZATION -----			
IOP	ACTIVITY RATE	AVG Q LNTH	% IOP BUSY	% CMPR BUSY	I/O START RATE	INTERRUPT RATE
SYS	29025.50	0.00	0.92	12.01	27745.45	30082.68

Figure 3-29 RMF report for the Integrated Accelerator for zEDC on IBM z15

### 3.8.2 Conclusion

The Integrated Accelerator for zEDC on IBM z15 improves the compression ratio across well-known file types such as TXT, PDF, JPG, PNG, PPT, and TIF. It can achieve up to 70% faster compression and decompression operations in some cases. Overall, compression time is reduced by 56% - 77% for insert operations, and decompression time by 13.5% - 75% for select operations on LOB data. The Integrated Accelerator for zEDC also reduces CPU cost and can handle much more compression work compression than the PCIe card on IBM z14 hardware.

In general, when you can achieve a good compression ratio (better disk space saving), you tend to observe both a CPU and elapsed time reduction by compressing the LOB. However, with low compression ratio files, you might experience some regression for both elapsed and CPU time when you enable LOB compression.

## 3.9 Asynchronous cross-invalidation: IBM z14 and later

Cross-invalidation (XI) is done through Write-And-Register (WAR) (**IXLCACHE REQUEST(WRITE\_DATA)**) or Write-And-Register-Multiple (WARM) (**IXLCACHE REQUEST(WRITE\_DATA\_LIST)**) commands, which are MVS services. Specifically, WAR is for a single page that is written to the GBP, and WARM is for multiple pages that are written to GBP in a single operation.

Before this feature, both WAR and WARM ran synchronously, which means that Db2 stays with the command from the time that it is issued until it completes. The operation includes the data being written to the GBP and XI signals traveling to the rest of data-sharing members to invalidate local buffers. Only when both operations are complete can Db2, which issued this command, proceed to its next task.

For a sysplex over distance, when the CFs and LPARs are separated by long distance (order of kilometers), these commands can take a while to complete due to the XI signals traveling a long distance.

With the asynchronous cross-invalidation feature, Db2 gets control back as soon as the WAR or WARM command is issued. To avoid data inconsistency, Db2 issues a sync-up call to check whether the commands completed in the meantime. Only the last token must be checked. If the command is not yet completed when it is checked, Db2 is suspended until it completes. This behavior allows multiple asynchronous WAR or WARM commands to run simultaneously, which results in performance improvements.

### 3.9.1 Requirements

To leverage the asynchronous cross-invalidation feature, you must be using the following environment:

- ▶ IBM z14.
- ▶ GBP allocated in a CF on IBM z14 and CF Level 23 or later
- ▶ z/OS 2.3 or later (or z/OS 2.2 with APAR OA54688)
- ▶ Db2 12 at FL 100 or later
- ▶ Db2 12 APAR PH10577

### 3.9.2 Performance measurement

The following three factors accounted for during the testing:

- ▶ The size of the page
- ▶ The number of the pages in the **WARM** command
- ▶ The distance between the CF and LPAR that the XI signal must travel

Different combinations of these factors were tested by using batch jobs to determine the threshold in which asynchronous processing of the XI request outperforms a synchronous call. Update commit suspend time is the criteria that we used to determine the threshold. Under that threshold, synchronous calls show shorter suspend times; however, above the threshold, asynchronous calls show shorter suspend times, which provide a performance benefit.

Table 3-13 on page 81 shows the update commit suspend times for two different distances between the CF and LPARs (local and 10 km apart) from a test that uses a 4 K page size.



Table 3-13 Update commit time by using synchronous and asynchronous XI

Update commit suspend time (ms.)	Distance between CF and LPARs	Synchronous	Asynchronous
X pages/WARM	Local/Minimal	0.467	0.415
X pages/WARM	10 km	1.908	1.029

The data shows that for a 4 K page size, when the number of pages in a WARM command is below the internal threshold of X pages, Db2 issues the WARM command in synchronous fashion, which is the same behavior as without asynchronous cross invalidation. If the number is X or greater, Db2 issues the WARM command asynchronously regardless of the distance between CF and MVS LPARs, which results in better performance.

We repeated this exercise for 8 K, 16 K, and 32 K page sizes to determine the threshold value for the respective page sizes. Unfortunately, no threshold value can be found for a 32 K page size. So, Db2 does not allow the use of asynchronous XI for cross-invalidation of 32 K pages. This limitation is documented in APAR PH10577.

In addition to single-thread performance experiments, multi-thread test coverage includes the following observations:

- ▶ For multi-thread batch update jobs, class 2 elapsed time was improved.
- ▶ For multi-thread batch insert jobs, update commit suspend time was improved when many rows are inserted within a commit scope.
- ▶ No regression was observed when only a few rows were inserted within a commit scope.
- ▶ No regression was observed for our IBM CICS® to Db2 Online Transaction Processing (OLTP) workload that uses two-way data sharing with no distance between CF and LPARs.

Table 3-14 illustrates the performance improvement of using asynchronous XI when CF and LPARs are a minimal distance apart. These results showcase batch updating eight 4 K pages in a commit scope.

Table 3-14 Batch update benefit by using asynchronous XI without distance

Without distance	Synchronous XI	Asynchronous XI	Delta = Async. XI/Sync. XI
Class 2 elapsed time (sec)	384.52	362.63	-5.7%
Class 2 CPU time (sec)	105.91	105.93	+0.0%

Table 3-15 illustrates the performance improvement of using asynchronous XI when CF and LPARs are 10 km apart. These results showcase batch updating eight 4 K pages in a commit scope. The same batch jobs that were used for the test in Table 3-14 are used in this test.

Table 3-15 Batch update benefit by using asynchronous XI with 10 km distance

Distance of 10 km	Synchronous XI	Asynchronous XI	Delta = Async. XI/Sync. XI
Class 2 elapsed time (sec)	1161.23	914.39	-21.2%
Class 2 CPU time (sec)	101.54	105.22	+3.6%

### 3.9.3 Monitoring information

A new DSNB818I message is issued by the **-DISPLAY GROUPBUFFERPOOL** command, which includes the **MDETAIL** option when asynchronous cross-invalidations were performed, as shown in the following example:

```
DSNB818I  -DBCL ASYNCHRONOUS CROSS-INVALIDATION
           WRITES WITH ASYCH XI           = 279075
           SYNCH-UP CALLS                 = 279065
           SUSPENSIONS FOR SYNCH-UP       = 136744
```

Fields QBGLWX, QBGLSU, and QBGLAS were added to the Db2 statistics trace record IFCID 2 to make this information available for regular subsystem performance monitoring.

### 3.9.4 Usage considerations

This feature is transparent to the user. No action is needed to activate it.

### 3.9.5 Conclusion

The asynchronous cross-invalidation feature addresses the elongated CF response time in a sysplex where a long distance is involved. By invoking the asynchronous cross-invalidation feature, when it is appropriate, update commit suspend time is reduced, which results in improved elapsed time without sacrificing CPU time. The threshold to trigger the asynchronous cross-invalidation feature depends on the page size. Through a series of experiments, a threshold was found for 4 K, 8 K, and 16 K page sizes, but not the 32 K page size. The internally designated threshold values are transparent to users, and 32 K pages are not eligible for asynchronous cross-invalidation calls. When the number of changed pages that is written to GBP is below the threshold value, Db2 continues to issue calls that use synchronous XI, so no regression occurs.

The sync-up call that follows an asynchronous cross-invalidation call drives more requests to the GBP. That cost is reflected in Db2 MSTR SRB time, which is not zIIP eligible. The cost of the sync-up call might be noticeable with a remote CF.

Overall, the goal of asynchronous cross-invalidation, which is improving batch update elapsed time by sending “write and cross-invalidation” asynchronously, is achieved.

## 3.10 zHyperLink I/O

IBM zHyperLink is a direct short-distance link between IBM z14 and later systems and IBM storage controllers. It is designed to achieve low I/O latency by directly connecting the IBM zSystems central processor complex (CPC) to the I/O bay of the IBM storage controller. The physical link can be up to 150 meters, as shown in Figure 3-30 on page 84. The low I/O latency of zHyperLink, which can be up to 10 times faster than the latest 16 G IBM Z High-Performance FICON (zHPF) channel, makes it suitable for implementing true synchronous I/O. In the synchronous I/O implementation, the CPU waits or spins for the I/O operation to complete.

Traditionally, I/O is processed asynchronously regarding CPU. Asynchronous I/O involves dispatching, interrupt handling, CPU queuing, loading, and reloading the processor cache. In a loaded busy system, these different steps that are associated with asynchronous I/O can elongate the I/O response time. Because zHyperLink is a direct, high-speed link between the CPC and the IBM storage system, the CPU can wait for the I/O to complete, making the I/O operation truly synchronous to the CPU. Therefore, the CPU cost of dispatching and interrupt handling is no longer necessary, and the Db2 code is not evicted from processor cache. The zHyperLink feature eliminates all delays that are associated with asynchronous I/O.

Enterprise systems are increasingly challenged to meet service-level agreements (SLA) to handle the growth from the cloud and mobile application stacks. It is even more challenging as new data sources are added. Therefore, it became important that a Db2 system can keep up with the demand for fast transaction response times. This change put increasing pressure on Db2 database administrators to fine-tune systems to meet the response time requirements. The most straightforward way of improving response time is to avoid I/O. By using Db2 in-memory buffer pools, Db2 transactions can avoid I/O operations. To do so requires increasing the Db2 buffer pool size, which might require purchasing larger quantities of memory to hold the tables that are being accessed.

When I/O cannot be avoided for response time-critical applications, Db2 administrators find it hard to achieve the wanted transaction response times due to high I/O latency. In such cases, transaction elapsed time can be drastically reduced by Db2 leveraging IBM zHyperLink technology for I/O without changing the application. Db2 leverages zHyperLink for database synchronous random read I/O and active log force write I/O, in which Db2 is suspended until I/O completion.

During zHyperLink I/O, the CPU spins while waiting for the I/O to complete. The CPU spin is a fixed clock time. In IBM zSystems hardware with higher clock speed processors, the fixed reduced I/O latency time corresponds to a higher number of CPU cycles. As a result, the drastically reduced I/O response time for zHyperLink comes with higher CPU cost for IBM zSystems with faster processors. However, in a busy system where traditional I/O is impacted by delays that are associated with asynchronous I/O, the CPU increase with zHyperlink might be acceptable for achieving substantially reduced transaction elapsed time for critical applications.

Db2 leverages zHyperLink technology for both database synchronous random read I/O and active log force write I/O.

Figure 3-30 shows a zHyperLink high-level configuration.

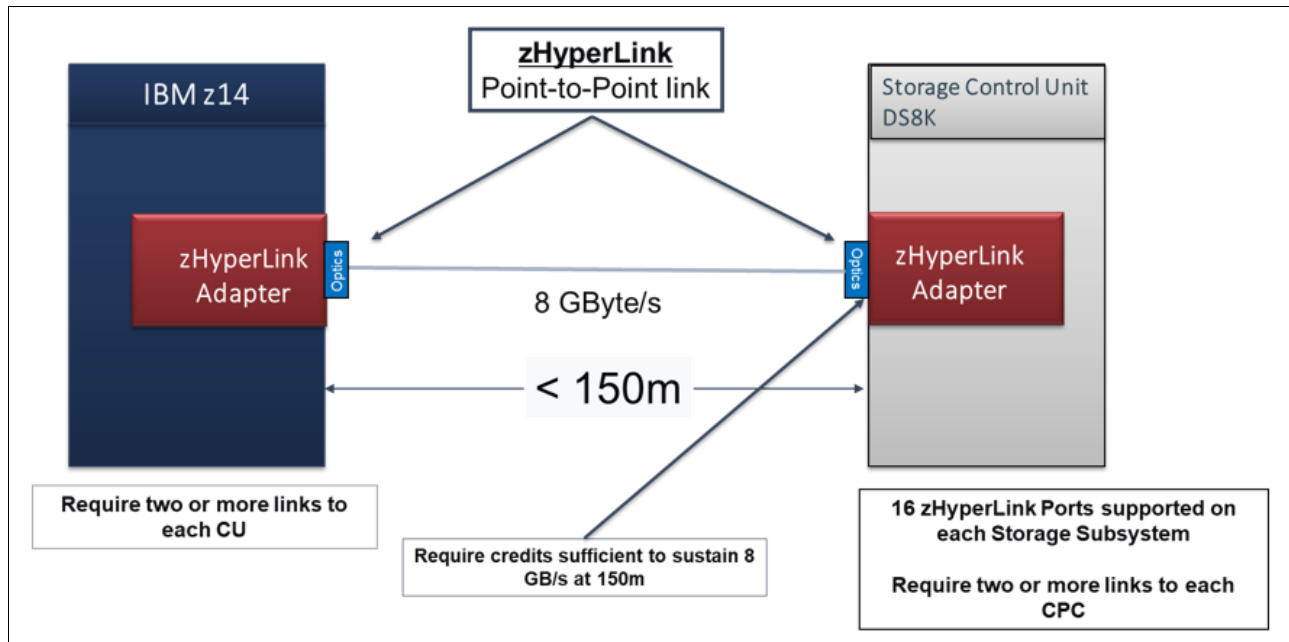


Figure 3-30 zHyperLink high-level configuration

### 3.10.1 How to enable zHyperLink I/O for Db2

Enabling zHyperLink technology on a z/OS LPAR requires both hardware and software updates.

To identify all hardware and software requirements, see “zHyperLink Prerequisites” in *Getting Started with IBM zHyperLink for z/OS*, REDP-5493. PTFs for zHyperLink can be found by searching for the IBM.Function.zHyperLink fix category with APAR keyword HYPERL/K.

In order for Db2 systems to successfully run zHyperLink I/O, zHyperLink must be enabled at the z/OS LPAR level, the storage class level, and the Db2 level.

#### z/OS LPAR level

ZHYPERLINK is disabled by default at the z/OS LPAR level:

- ▶ To enable zHyperLink during an IPL of the LPAR, add the following statements in the IECIOSxx PARMLIB member:
  - ZHYPERLINK,OPER=READ (enables zHyperLink for READ)
  - ZHYPERLINK,OPER=WRITE (enables zHyperLink for WRITE only)
  - ZHYPERLINK,OPER=ALL (enables zHyperLink for both READ/WRITE)
 ZHPF=YES is a prerequisite for enabling ZHYPERLINK.
- ▶ To dynamically enable zHyperLink, issue the SETIOS MVS command from the operator console:
  - SETIOS ZHYPERLINK, OPER=READ (enables zHyperLink for READ only)
  - SETIOS ZHYPERLINK, OPER=WRITE (enables zHyperLink for WRITE only)
  - SETIOS ZHYPERLINK, OPER=ALL (enables zHyperLink for both READ/WRITE)

- ▶ To dynamically disable zHyperLink, issue the **SETIOS MVS** command:  
SETIOS ZHYPERLINK, OPER=NONE (disables zHyperLink)
- ▶ To display the status of zHyperLink at the LPAR level, issue the following command:  
D IOS, ZHYPERLINK
- ▶ Because zHPF is a prerequisite for zHyperLink, verify that zHPF is enabled by issuing the following command:  
D IOS, ZHPF
- ▶ If zHPF is not enabled, enable zHPF by issuing the following command:  
SETIOS ZHPF=YES

### Storage class level

The SMS storage class of the Db2 data sets must be enabled for zHyperLink eligibility by using the Integrated Storage Management Facility (ISMF).

**Note:** The default behavior is not to use ZHYPERLINK for the SMS storage class.

- ▶ To enable zHyperLink for read I/O operations, set zHyperLink Eligible for Read to YES.
- ▶ To enable zHyperLink for write I/O operations, set zHyperLink Eligible for Write to YES.

These fields are shown in Figure 3-31.

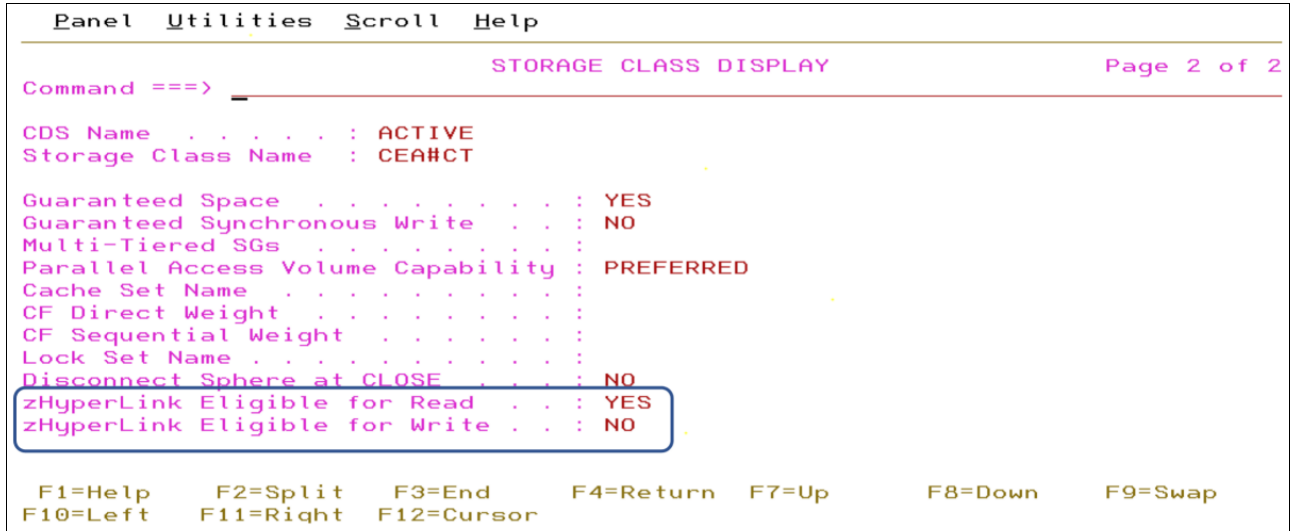


Figure 3-31 Integrated Storage Manager Facility STORAGE CLASS DISPLAY

### Db2 level

Use the Db2 **ZHYPERLINK** subsystem parameter to enable zHyperLink at the Db2 level. The new **ZHYPERLINK** subsystem parameter can be updated online. The default setting is DISABLE.

Set **ZHYPERLINK** to one of the following three options, which control the scope of the zHyperLink protocol for I/O requests:

- ▶ ZHYPERLINK=DATABASE (enables zHyperLink for database read only)
- ▶ ZHYPERLINK=ACTIVELOG (enables zHyperLink for Active Log Write I/O only)
- ▶ ZHYPERLINK=ENABLE (enables zHyperLink for both database read I/O and Active Log Write I/O)

After you enable zHyperLink at all three levels, you can issue the following commands to determine whether zHyperLink is used:

- ▶ To display the status of zHyperLink at the LPAR level, issue the following command:  
D IOS, ZHYPERLINK
- ▶ To display whether zHyperLink is enabled for a device or to display the reasons why it is disabled, issue the following command:  
D M=DEV(device\_number),ZHL (or ZHYPERLINK)
- ▶ To display information for a specific PCIe Function Identifier (PFID), PFID or PFID=ALL can be specified to get information for all PFIDs. If you specify DETAIL, it shows the number of zHyperLink successes, link busies, and timeouts, for example:  
D IOS,ZHL,PFID=pfid[,DETAIL]
- ▶ To display information at the link level, LINK or LINK=ALL can be specified to get information for all links. DETAIL provides detailed PFID information for that link, for example:  
D IOS,ZHL,LINK=pchid.port[,DETAIL]
- ▶ To display the ZHL response code or qualifier history of any control unit number on IBM DS8000®, issue the following command:  
D IOS,ZHL,DIAG,CU=cunum

### 3.10.2 zHyperLink and database read I/O

zHyperLink support for read is leveraged by Db2 11 (with the PTF for APAR PH01123 applied) and later for synchronous database random read I/O. The synchronous read I/O that uses IBM zHyperLink occurs only if the requested data is in the IBM storage control unit cache. If the data is not found in the cache, the host is notified, and traditional asynchronous I/O runs on an zHPF channel. Therefore, IBM zHyperLink technology works with the existing zHPF channels and enables the zHyperLink technology implementation to complement high-performance IBM FICON® channels.

During zHyperLink I/O, the CPU spins while waiting for the I/O to complete. zHyperLink can reduce the time that is required to complete the I/O because the major processing components for standard I/O operations, such as the dispatching, interrupt handling, and CPU queue time, are no longer necessary, and Db2 code is not evicted from the processor cache by other activities in the system. Concurrently, zHyperLink read I/O time is included as a part of Db2 class 2 CPU time (SQL processing time), which is like synchronous CF requests, such as GBP reads where the CPU time for CF access is charged to Db2 class 2 CPU time.

To estimate the benefit of zHyperLink, it is important to understand the disk cache hit rate. Db2 and DFSMS added the additional instrumentation to indicate whether a disk cache hit occurred or not.

To evaluate database read I/O eligibility for zHyperLink, Db2 added the following new fields in statistics and accounting records:

- ▶ SMF100 statistics record (Db2 subsystem)
  - The IFCID 2 trace record was updated with the QBSTSI0C field, which contains the number of database read I/O operations that resulted in disk cache hit when zHyperLink is not used.
- ▶ SMF101 accounting record (Db2 application)
  - The IFCID 3 trace record (Db2 plan-level accounting) was updated with the following fields:
    - QWACAWCD contains the number of database read I/O operations that resulted in a disk cache hit when zHyperLink is not used.
    - QWACAWTD contains the wait time for database read I/O operations that resulted in a disk cache hit when zHyperLink is not used.
    - QBACIO0C contains the number of database read I/O operations that resulted in a disk cache hit when zHyperLink is not used.
- ▶ The IFCID 239 trace record (Db2 package-level accounting) was updated with the QBACIO0C field, which contains the number of database read I/O operations that resulted in a disk cache hit when zHyperLink is not used.
- ▶ DFSMS
  - DFSMS added data-set-level information to SMF 42 record type 5 and 6. The following new fields were added to the SMF 42 type 6 record that can be used for identifying eligible candidates for zHyperLink read I/O:
    - S42SNERD contains the number of read requests that are eligible for zHyperLink I/O but were not attempted.
    - S42SNERH contains the number of read hits that are eligible for zHyperLink I/O but were not attempted.

The new fields are also available in the SMF42 type 5 record. APAR PI87082 must be installed in Db2 12 or APAR PI99235 in Db2 11 to identify eligible candidates for synchronous reads. The DFSMS APARs that are listed in Table 3-16 should also be installed for the zHyperLink SMF42 counters to take effect.

*Table 3-16 DFSMS APARs that are required for zHyperLink SMF42 counters*

APAR	DFSMS	HDZ2210	HDZ2220	HDZ2230
OA53889	Media Manager	UA94433	UA94436	UA94437
OA53963	Dev Support	UA94429	UA94430	UA94431
OA54112	CMM	UA94438	UA94439	UA94440

The IBM zSystems Batch Network Analyzer (zBNA) tool can be used to identify eligible synchronous read candidates for zHyperLink. This tool can be downloaded from [IBM Z Batch Network Analyzer \(zBNA\) Tool](#). The tool provides a top candidate list for zHyperLink and estimates the benefit of zHyperLink I/O. Before SMF42 type 6 records for zBNA zHyperLink eligibility analysis can be collected, the new Db2 **ZHYPERLINK** subsystem parameter must be set to DISABLE or the zHyperLink feature must be disabled at the input/output supervisor (IOS) level. The default setting of the **ZHYPERLINK** subsystem parameter in Db2 is DISABLE.

An example of the zBNA top data set candidate list for zHyperLink is shown in Figure 3-32.

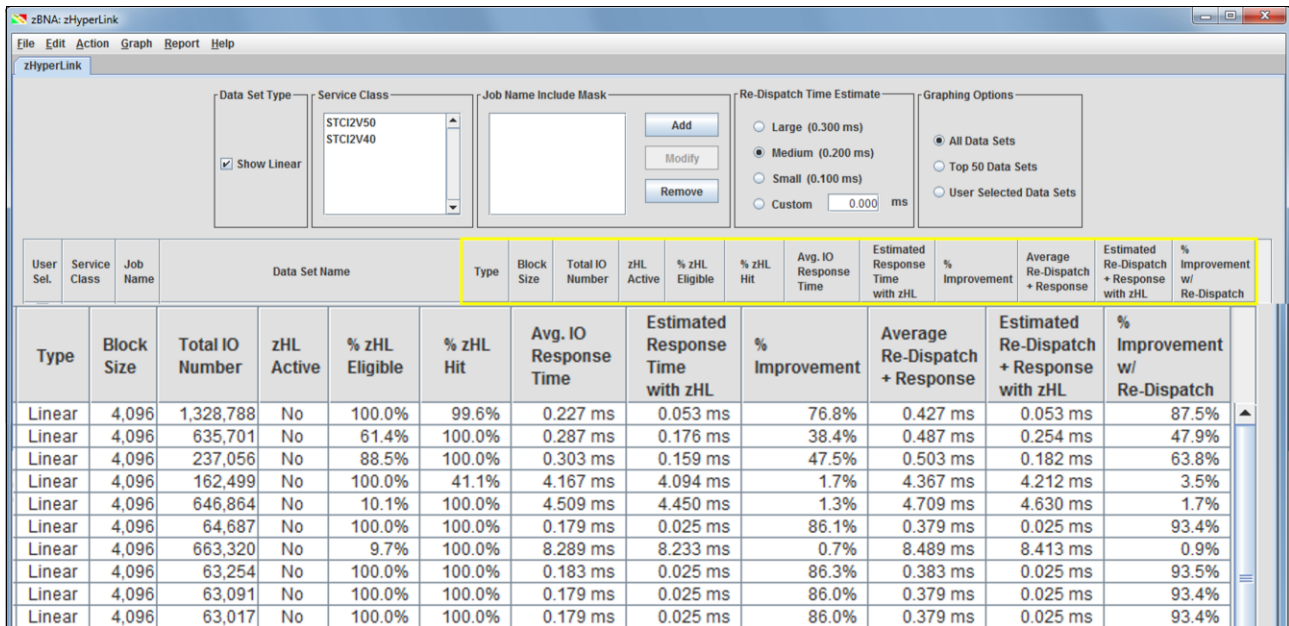


Figure 3-32 zBNA top data set candidates for zHyperLink read I/O

An example of the zBNA Data Filtering Capability report is shown in Figure 3-33.

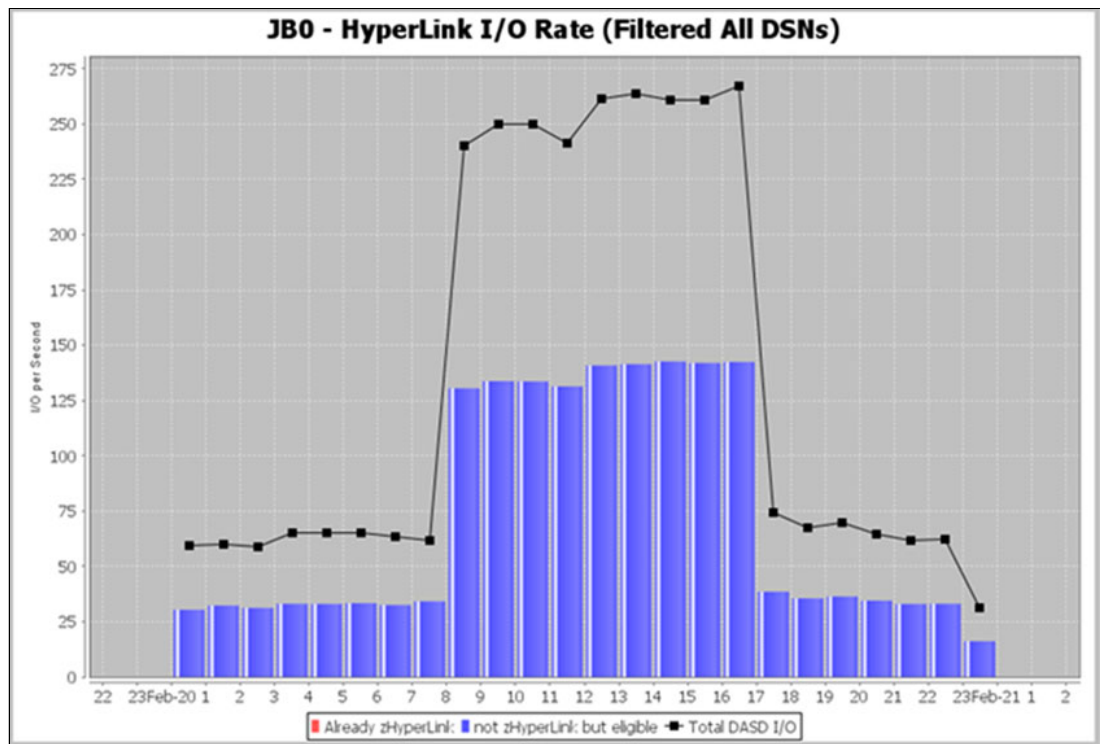


Figure 3-33 zBNA total I/O rate and zHyperLink eligible I/O rate



An example of the zBNA Estimated zHyperLink I/O Response Time report is shown in Figure 3-34.

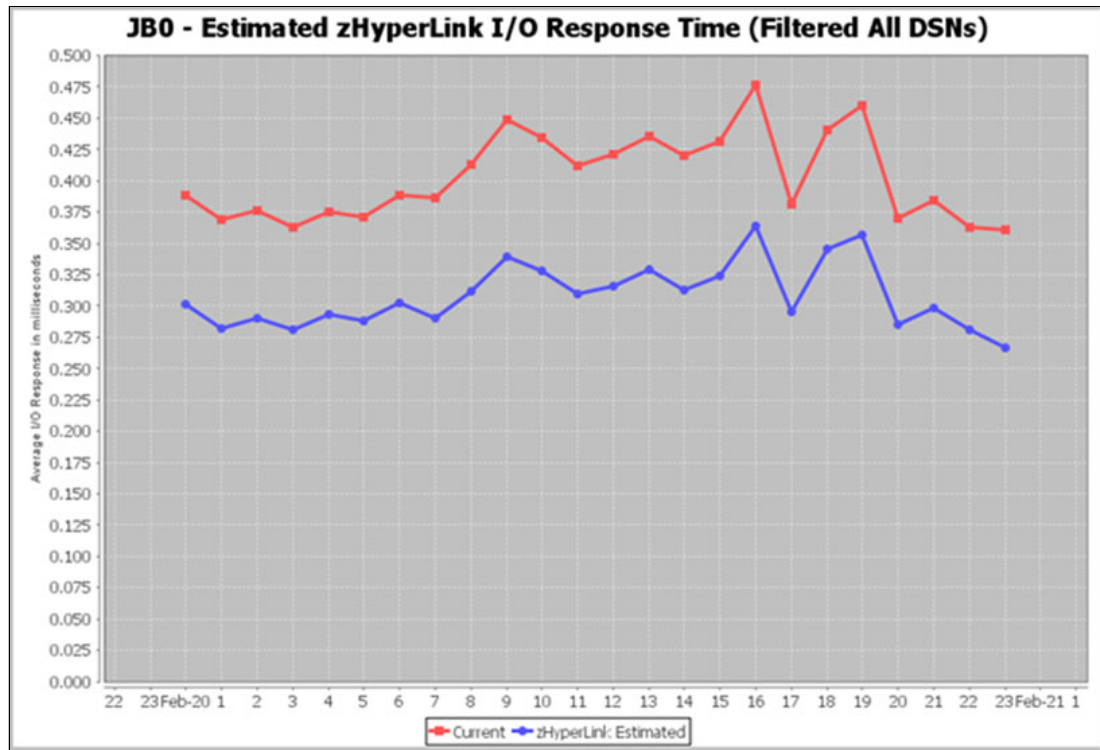


Figure 3-34 zBNA estimated zHyperLink I/O response time

### 3.10.3 Requirements

To leverage zHyperLink for Db2 database read I/O, you must be using the following environment:

- ▶ IBM z14 or later
- ▶ IBM DS8880 or later storage subsystem
- ▶ z/OS 2.2 or later (You can find the PTFs for zHyperLink by searching for Fix Category IBM.Function.zHyperLink with APAR keyword HYPERL/K.)
- ▶ Db2 11 (with APAR PH01123) or later

### 3.10.4 Performance measurements

A number of performance measurements were run by using the IBM brokerage transaction workload, which generates many random database read I/Os.

#### Measurement environment

Performance measurements for zHyperLink reads were performed by using the following environment:

- ▶ IBM z14 M05-3907-7E7 and z14 zR1 connected to IBM DS8886
- ▶ z/OS 2.2 with the necessary maintenance

- ▶ Db2 12 at FL 501
- ▶ IBM brokerage transaction workload (approximately an 800 GB database)
 

The transactions ran as native SQLPL applications from Linux for IBM Z clients, with approximately 4000 transactions per second both with and without extra, heavy query workloads running in the background. Most of the measurements were done with less than a 95% disk cache hit rate.

### Measurement scenario description

We ran the following sets of measurements to evaluate the performance of the IBM brokerage transaction workload when zHyperLink is used for database random read I/O and compared them to database read I/O that uses zHPF links:

- ▶ An IBM brokerage transaction workload with two different buffer pool sizes to vary the number of database random I/Os and evaluate transaction elapsed time.
- ▶ An IBM brokerage transaction workload with two different buffer pool sizes to vary the number of database random I/Os and evaluate CPU usage.
- ▶ An IBM brokerage transaction workload on IBM z14 processors with different processor speeds to evaluate the impact of processor speed on CPU usage.

### Results

The first set of measurements illustrates the benefit of reduced transaction elapsed time with zHyperLink enabled for database read I/O, and shows that the more database I/Os that are DASD cache hits, the larger the reduction in elapsed time.

Figure 3-35 compares the average transaction elapsed time for the same workload, one with zHyperLink enabled for database read I/O and the other with standard I/Os that use high-performance FICON. The total size of buffer pools is set as 10 GB, and there are an average of 28 random database read I/Os per transaction.

Because the database read I/O wait time is reduced dramatically with zHyperLink, the total elapsed time is reduced by half.

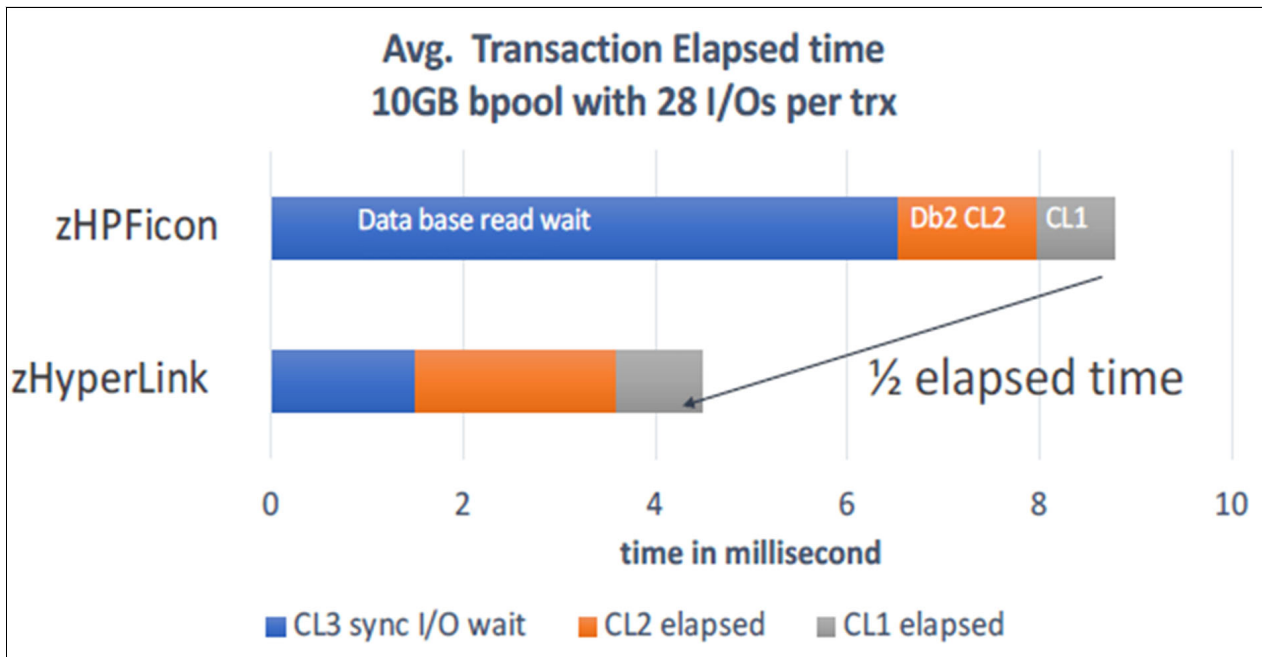


Figure 3-35 Average elapsed time with a 10 GB bpool with zHyperLink read

Next, the same workload was run with a total size of buffer pools size of 70 GB, and there are an average of 7.5 random database read I/Os per transaction.

Again, due to the reduction of the database read I/O wait time, the transaction elapsed time is reduced by 23% with zHyperLink, as shown in Figure 3-36. The benefit is less than the 10 GB buffer pool test because there are fewer database I/Os due to the higher buffer pool hit ratio with larger buffer pools.

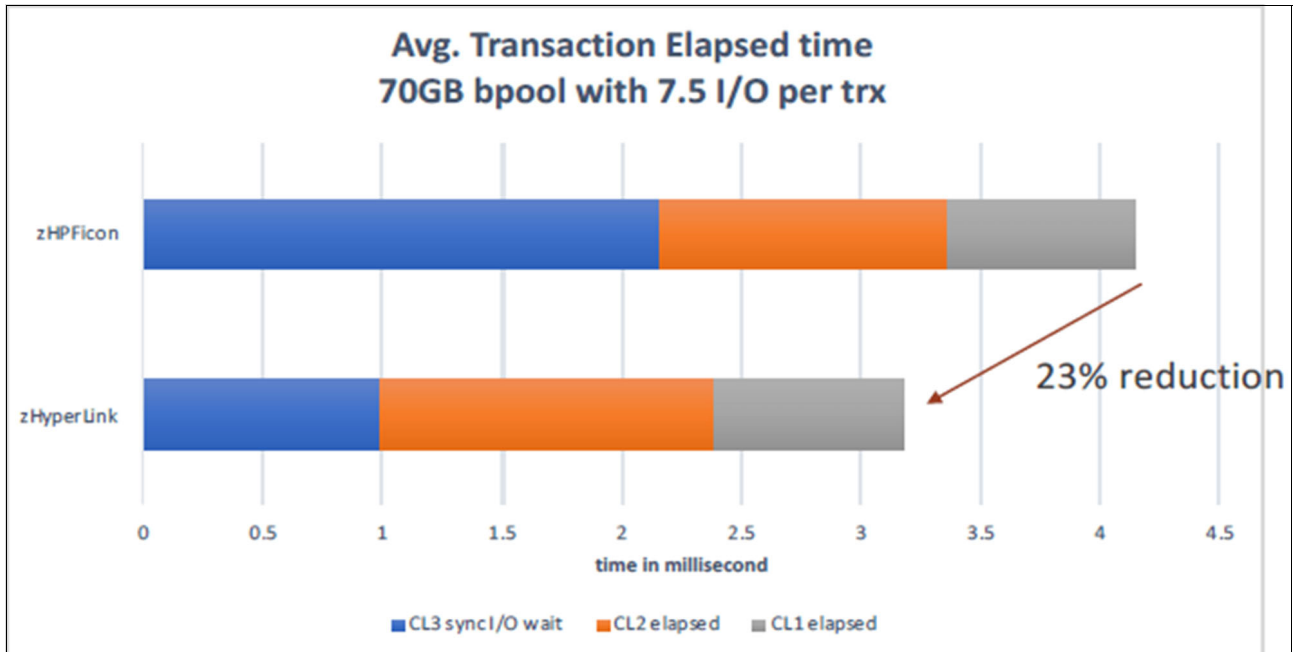


Figure 3-36 Average elapsed time with a 70 GB bpool with zHyperLink read

zHyperLink I/O latency is a fixed wall clock time. Although the I/O latency benefit is same, the zHyperLink CPU cost depends on the following factors:

- ▶ The speed of processors: The faster the processors, the faster they spin. Therefore, waiting for I/O completion becomes more expensive.
- ▶ The path reduction and processor cache benefit: Path length savings are a result of eliminating I/O interrupt processing and dispatching delay, and not missing the processor cache during I/O operation. Although the path length savings are consistent, the benefit from cache miss depends on the relative nesting intensity (RNI) of your workloads (see [LSPR workload categories](#)).

The tradeoff between the CPU costs of zHyperLink Read relative to the processor speed is shown in Figure 3-37.

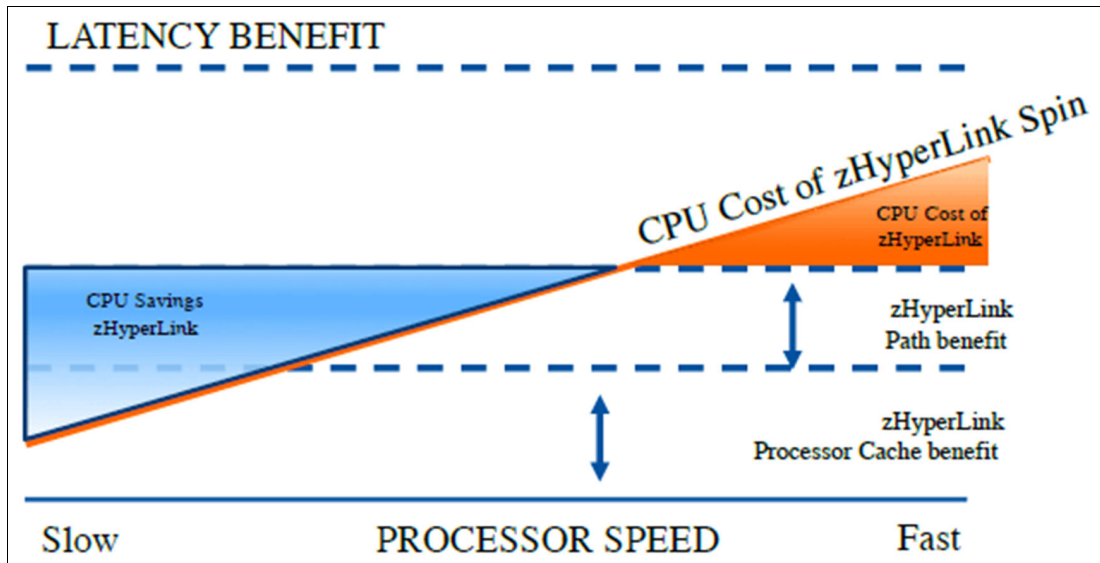


Figure 3-37 CPU cost of zHyperLink reads relative to the processor speed

The next set of measurements demonstrates CPU usage when zHyperLink is used for database read I/O.

With 70 GB buffer pools, we observed a 5% CPU increase at the LPAR level and a 13.8% CPU increase with a 10 GB buffer pool size. Most of the cost is in Db2 class 2 time. Because there are more cache hits in the Db2 buffer pool and less zHyperLink I/Os with 70 GB buffer pools compared to 10 GB buffer pools, we observe less elapsed time benefit, and less CPU impact.

The measurements in Figure 3-38 on page 93 were done on the LPAR where we ran only the Db2 transaction workload.

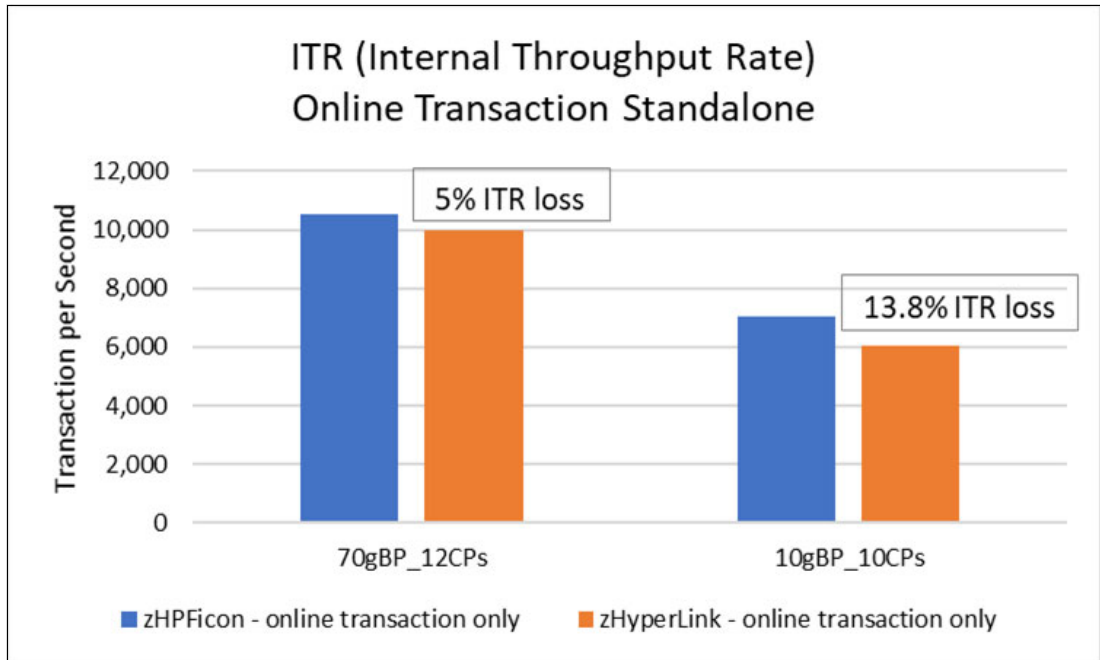


Figure 3-38 ITR of online transaction without a background workload

Although the measurement results in Figure 3-38 are clean, lab-controlled results, the next set of measurements are likely more realistic and better reflect real-world Db2 application usage. For the measurements that are shown in Figure 3-39, we ran several background Db2 applications, such as Db2 batch queries, in addition to the online transaction workload.

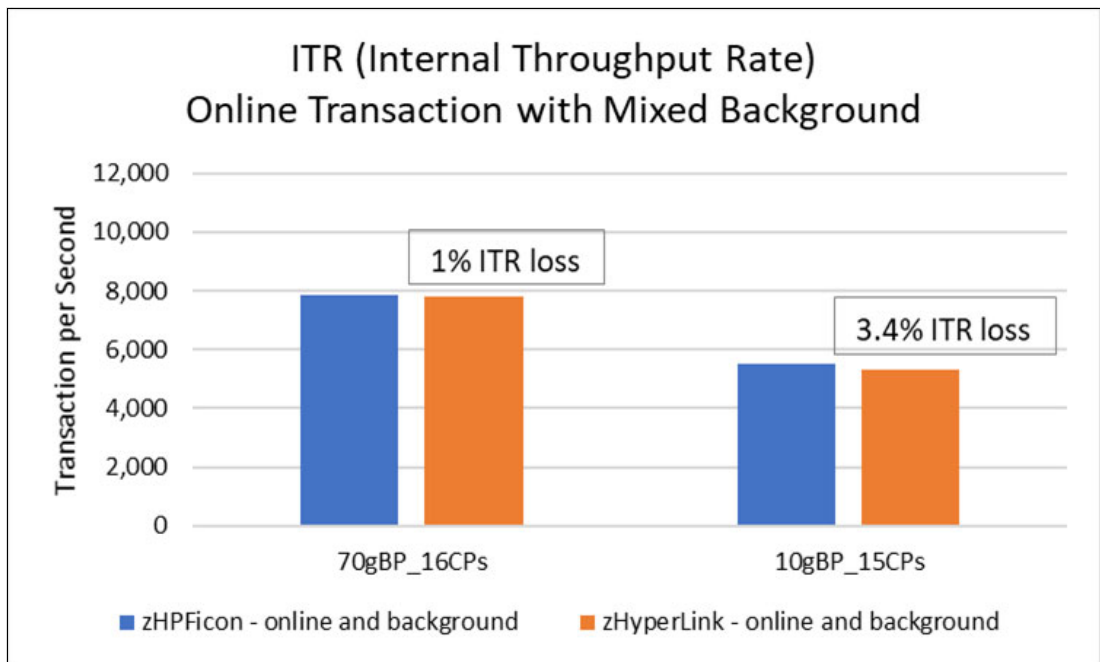


Figure 3-39 ITR of online transaction with a background workload

With the background workload running, we saw a smaller overhead (1 - 3.4% CPU impact) for the online transaction workload with zHyperLink enabled for database read I/O. This result is likely due to more disruptions in the processor cache with the background workloads running, and the cost of standard I/Os are more expensive than in a pristine environment.

The last set of measurements demonstrates the impact of processor speed on CPU usage. These measurements were taken by using IBM z14 Model ZR1 processors with a lower CPU cycle speed and by using similar workloads but at a smaller transaction rate because the processor speed is reduced compared to a full-speed IBM z14 system. Interestingly, as we lower the processor speed from ZR1 Z06 to ZR1 I06, the CPU impact of zHyperLink shifts from negative to positive, as shown in Figure 3-40. This result clearly illustrates the points that CPU savings and cost cross at some point, and that cost depends on many factors.

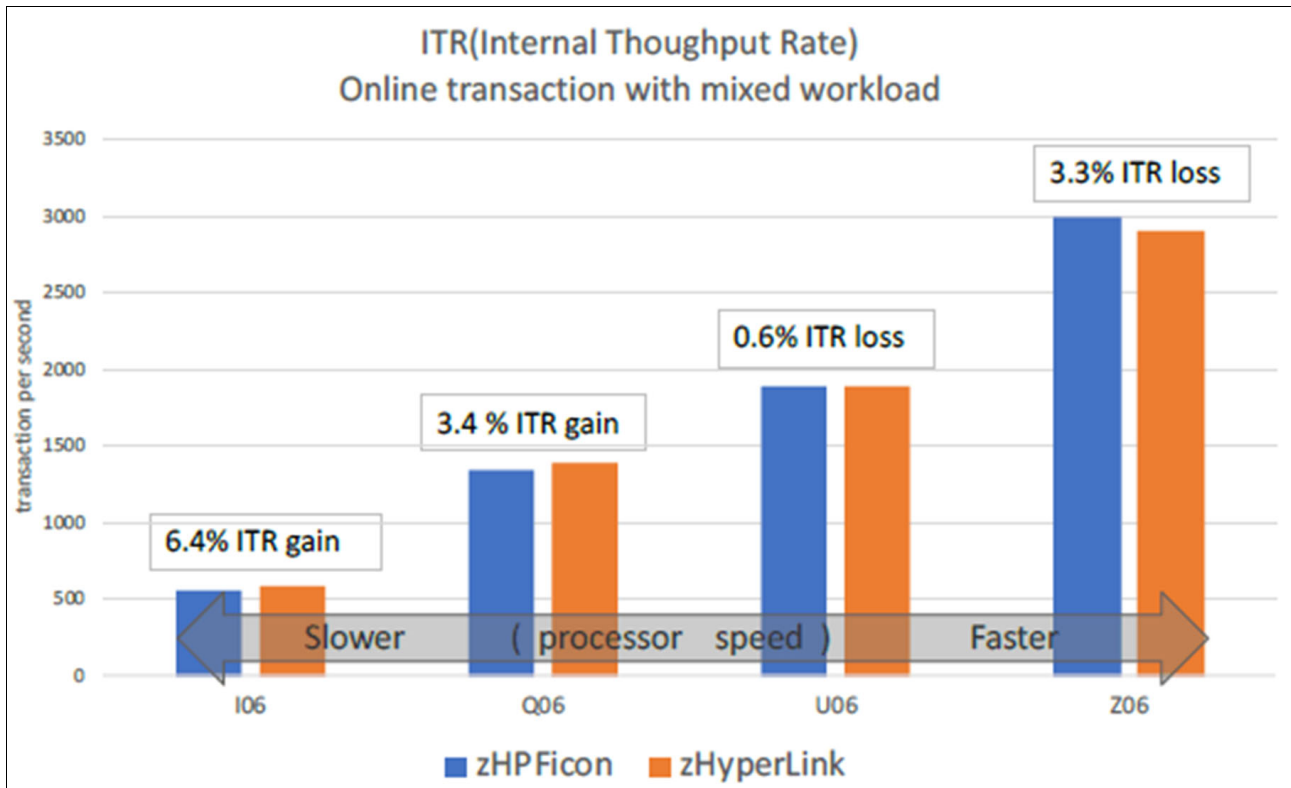


Figure 3-40 ITR with background workload on different ZR1 models

### Summary of results

These sets of measurements that use the IBM brokerage transaction workload demonstrate that the average transaction response time can be reduced up to 50% when zHyperLink is used for database random read I/O compared to using zHPF. The elapsed time benefit depends on the number of database random read I/Os that are eligible for zHyperlink. The CPU impact that is shown as ITR loss might not be substantial depending on processor speed, and with a slow processor speed, the ITR gain can be up to 6.4%. The CPU impact also can be less when other background work is running on the LPAR.

### 3.10.5 Monitoring Db2 zHyperLink database I/O

Performance information on zHyperLink read I/O is collected by Db2 and z/OS.

#### Db2 monitoring of zHyperLink database read I/O

Db2 added several new fields to accounting and statistics records to monitor zHyperLink I/O:

▶ SMF100 statistics record

The IFCID 2 trace record (Buffer Manager section) was updated with the QBSTSYI0 field, which is the number of successful read I/O operations that are using zHyperLink.

▶ SMF101 accounting record

zHyperLink I/O spin time is accounted for in accounting class 2 and class 7 CPU time. The IFCID 3 and 239 trace record (Buffer Pool section) were updated with the following fields:

- QBACSYI contains the number of successful read I/O operations that used zHyperLink.
- QBACSYIT contains the elapsed time for successful read I/O operations that used zHyperLink.

The class 3 and class 8 Db2 synchronous database read/wait time is only for traditional I/O and not zHyperLink read I/O.

▶ SMF102 performance record

The IFCID 199 records for data set I/O statistics by buffer pool. The following new fields were added:

- QW0199ZP contains the total synchronous zHyperLink pages.
- QW0199ZV contains the average I/O delay, in milliseconds, for synchronous read I/O that used zHyperLink.
- QW0199ZX contains the maximum I/O delay, in milliseconds, for synchronous read I/O that used zHyperLink.
- QW0199Z1 contains the average I/O delay, in microseconds, for synchronous read I/O that used zHyperLink.
- QW0199Z2 contains the maximum I/O delay, in microseconds, for synchronous read I/O that used zHyperLink.

IFCID 6 and 7 trace records for begin read I/O and end read I/O, respectively. The following new fields were added:

- QW0006SI X'40' indicates whether zHyperLink processing was used:
  - 1 = Read request with zHyperLink processing.
  - 0 = Read request without zHyperLink processing.
- QW0007SI X'40' indicates the status of zHyperLink processing:
  - 1 = Read request with zHyperLink processing was successful.
  - 0 = Read request without zHyperLink processing was suspended.



► The **DISPLAY** command

The **-DISPLAY BUFFERPOOL LSTATS** command shows the total number of zHyperLink read I/O operations (Figure 3-41) and the average zHyperLink read I/O delay times under SYNCHRONOUS I/O at the page set level (Figure 3-42).

```

DSNB409I -CEA2 INCREMENTAL STATISTICS SINCE 15:49:08 JUL 14, 2022
DSNB411I -CEA2 RANDOM GETPAGE      =1320354448
          SYNC READ I/O (R) =336337
          SEQ.   GETPAGE   =172400
          SYNC READ I/O (S) =16
          SYNC READ I/O (ZHL) =304983
          DMTH HIT          =0
          PAGE-INS REQUIRED =0
          SEQUENTIAL       =4326
          VPSEQT HIT       =0
          RECLASSIFY       =4261
DSNB412I -CEA2 SEQUENTIAL PREFETCH -
          REQUESTS         =0
          PREFETCH I/O     =0
          PAGES READ       =0
DSNB413I -CEA2 LIST PREFETCH -
          REQUESTS         =59
          PREFETCH I/O     =4
          PAGES READ       =53
DSNB414I -CEA2 DYNAMIC PREFETCH -
          REQUESTS         =45347
          PREFETCH I/O     =423
          PAGES READ       =4231
  
```

Figure 3-41 **DISPLAY BPOOL** with zHyperLink I/O at the buffer pool level

```

-----PAGE SET/PARTITION STATISTICS-----
DSNB467I -CEA2  STATISTICS FOR INDEX SPACE TPCEA100.TETXIX1
INSTANCE 1 -
          DATA SET #:      1 USE COUNT:      0
DSNB453I -CEA2  VP CACHED PAGES -
          CURRENT           =   4   MAX           =   4
          CHANGED          =   0   MAX           =   0
DSNB455I -CEA2  SYNCHRONOUS I/O DELAYS -
          AVERAGE DELAY =1061   MAXIMUM DELAY   =4170
          TOTAL PAGES    =4
          SYNCHRONOUS I/O DELAYS WITH ZHYPERLINK -
          AVERAGE DELAY =24   MAXIMUM DELAY   =26
          TOTAL PAGES    =3
  
```

Figure 3-42 **DISPLAY BPOOL** with zHyperLink I/O at the page set level

**z/OS monitoring of zHyperLink**

z/OS also collects information about zHyperLink I/O at different levels. This information can be viewed in RMF reports.



**Device level (RMF device activity reports)**

The Direct Access Device Activity report, which is shown in Figure 3-43, shows only IBM FICON and zHPF (asynchronous I/O) activity. The S in the DEVICE ACTIVITY RATE field indicates that zHyperLink I/O requests were also performed for this device; therefore, the Synchronous I/O Device Activity report also should be checked.

D I R E C T   A C C E S S   D E V I C E   A C T I V I T Y															PAGE					
z/OS V2R5		SYSTEM ID SY41		START 07/14/2022-16.31.00		INTERVAL 000.05.59		END 07/14/2022-16.37.00		CYCLE 1.000 SECONDS		PAGE		1						
TOTAL SAMPLES = 360		IODF = 41		CR-DATE: 07/07/2022		CR-TIME: 12.00.17		ACT: ACTIVATE												
STORAGE GROUP	DEV NUM	DEVICE TYPE	NUMBER OF CYL	VOLUME SERIAL	PAV	LCU	DEVICE ACTIVITY RATE	AVG RESP TIME	AVG IOSQ	AVG CMR	AVG DB	AVG INT	AVG PEND	AVG DISC	AVG CONN	% DEV CONN	% DEV UTIL	% DEV RESV	AVG NMBR ALLOC	% ANY ALLOC
CEALGAB	0B000	3390A	99057	TAB000	1.0H	0090	1115.78	.152	.000	.000	.000	.000	.041	.000	.110	12.32	12.32	0.0	3.0	100.0
CEADBAB	0B001	3390A	99057	TAB001	1.0H	0090	29.5475	1.99	.018	.040	.000	.000	.119	1.48	.375	1.09	5.40	0.3	116	100.0
CEADBAB	0B002	3390A	99057	TAB002	1.0H	0090	39.3925	2.19	.000	.064	.000	.000	.156	1.55	.487	1.84	7.71	0.0	123	100.0
CEADBAB	0B003	3390A	99057	TAB003	1.0H	0090	24.3645	2.62	.000	.050	.000	.000	.140	2.01	.468	1.14	6.04	0.0	121	100.0
CEADBAB	0B004	3390A	99057	TAB004	1.0H	0090	99.7755	1.40	.000	.030	.000	.000	.094	1.04	.266	2.62	12.90	0.0	138	100.0
CEADBAB	0B005	3390A	99057	TAB005	1.0H	0090	82.9845	1.27	.000	.038	.000	.000	.107	.851	.314	2.61	9.67	0.0	127	100.0
CEADBAB	0B006	3390A	99057	TAB006	1.0H	0090	24.1895	2.07	.000	.039	.000	.002	.116	1.55	.407	0.97	4.66	0.0	115	100.0
CEADBAB	0B007	3390A	99057	TAB007	1.0H	0090	27.8005	2.50	.000	.072	.000	.001	.170	1.75	.577	1.60	6.47	0.0	120	100.0
CEADBAB	0B008	3390A	99057	TAB008	1.0H	0090	51.9395	1.94	.000	.038	.000	.002	.109	1.51	.319	1.66	9.50	0.0	138	100.0
CEADBAB	0B009	3390A	99057	TAB009	1.0H	0090	64.9975	1.74	.000	.045	.000	.000	.116	1.26	.363	2.30	10.25	0.0	113	100.0
CEADBAB	0B00A	3390A	99057	TAB00A	1.0H	0090	71.4145	1.42	.000	.033	.000	.000	.099	1.04	.286	2.01	9.34	0.0	122	100.0
CEADBAB	0B00B	3390A	99057	TAB00B	1.0H	0090	33.0505	2.32	.000	.038	.000	.001	.118	1.82	.380	1.26	7.28	0.0	116	100.0
CEADBAB	0B00C	3390A	99057	TBB00C	1.0H	0090	0.000	.000	.000	.000	.000	.000	.000	.000	.000	0.00	0.00	0.0	0.0	100.0

Figure 3-43 RMF Direct Access Device Activity report

The Synchronous I/O Device Activity report, which is shown in Figure 3-44, shows zHyperLink (Synch) I/O per second and average response times. More information about zHyperLink (synch) I/O activity, such as average transfer rate in megabytes, % request success, % link busy, % cache miss and % rejects, also is provided.

S Y N C H R O N O U S   I / O   D E V I C E   A C T I V I T Y															PAGE		
z/OS V2R5		SYSTEM ID SY41		START 07/14/2022-16.31.00		INTERVAL 000.05.59		END 07/14/2022-16.37.00		CYCLE 1.000 SECONDS		PAGE		1			
TOTAL SAMPLES = 360		IODF = 41		CR-DATE: 07/07/2022		CR-TIME: 12.00.17		ACT: ACTIVATE									
STORAGE GROUP	DEV NUM	DEVICE TYPE	VOLUME SERIAL	LCU	-- SYNCH I/O --	ASYNCH I/O	-- SYNCH I/O --	ASYNCH I/O	TRANSFER RATE	% REQ SUCCESS	% LINK BUSY	% CACHE MISS	-- REJECTS --	READ	WRITE		
CEADBAB	0B001	3390A	TAB001	0090	307.079	0.000	29.547	0.022	0.000	1.990	1.268	0.000	97.75	0.00	2.10	0.15	0.00
CEADBAB	0B002	3390A	TAB002	0090	409.521	0.000	39.392	0.022	0.000	2.193	1.691	0.000	97.44	0.00	2.46	0.10	0.00
CEADBAB	0B003	3390A	TAB003	0090	464.409	0.000	24.364	0.022	0.000	2.620	1.917	0.000	98.29	0.00	1.62	0.10	0.00
CEADBAB	0B004	3390A	TAB004	0090	566.090	0.000	99.775	0.022	0.000	1.405	2.340	0.000	96.81	0.00	3.11	0.08	0.00
CEADBAB	0B005	3390A	TAB005	0090	503.168	0.000	82.984	0.022	0.000	1.272	2.077	0.000	97.46	0.00	2.44	0.10	0.00
CEADBAB	0B006	3390A	TAB006	0090	396.356	0.000	24.189	0.022	0.000	2.068	1.636	0.000	98.20	0.00	1.70	0.10	0.00
CEADBAB	0B007	3390A	TAB007	0090	376.795	0.000	27.800	0.022	0.000	2.496	1.555	0.000	97.68	0.00	2.21	0.11	0.00
CEADBAB	0B008	3390A	TAB008	0090	521.374	0.000	51.939	0.021	0.000	1.937	2.152	0.000	97.40	0.00	2.53	0.07	0.00
CEADBAB	0B009	3390A	TAB009	0090	244.223	0.000	64.997	0.022	0.000	1.736	1.008	0.000	94.37	0.00	5.53	0.10	0.00
CEADBAB	0B00A	3390A	TAB00A	0090	442.626	0.000	71.414	0.022	0.000	1.424	1.827	0.000	97.08	0.00	2.83	0.09	0.00
CEADBAB	0B00B	3390A	TAB00B	0090	362.726	0.000	33.050	0.022	0.000	2.321	1.497	0.000	97.29	0.00	2.61	0.10	0.00
			LCU	0090	4595.17	0.000	549.451	0.022	0.000	1.758	18.97	0.000	97.33	0.00	2.57	0.10	0.00

Figure 3-44 RMF Synchronous I/O Device Activity report

### DASD subsystem level (RMF CACHE subsystem activity reports)

The Cache Subsystem Overview report, which is shown in Figure 3-45, shows zHyperLink I/O requests and the cache hits for the storage subsystem ID (SSID) in the SYNCH I/O ACTIVITY section.

CACHE SUBSYSTEM OVERVIEW														
TOTAL I/O	2348K	CACHE I/O	2348K											
TOTAL H/R	0.998	CACHE H/R	0.998											
CACHE I/O REQUESTS	COUNT	RATE	HITS	RATE	H/R	COUNT	RATE	FAST	RATE	HITS	RATE	H/R	%	
NORMAL	1722K	4796	1718K	4785	0.998	214344	597.1	214344	597.1	214344	597.1	1.000	88.9	
SEQUENTIAL	5144	14.3	5129	14.3	0.997	486810	1133	486810	1133	486810	1133	1.000	1.2	
CFW DATA	0	0.0	0	0.0	N/A	0	0.0	0	0.0	0	0.0	N/A	N/A	
TOTAL	1727K	4810	1723K	4799	0.998	621154	1730	621154	1730	621154	1730	1.000	73.5	
REQUESTS	READ	RATE	WRITE	RATE	TRACKS	RATE	COUNT		RATE					
NORMAL	4004	11.2	0	0.0	49608	138.2	DELATED DUE TO NVS		0		0.0			
SEQUENTIAL	15	0.0	0	0.0	530	1.5	DELATED DUE TO CACHE		0		0.0			
CFW DATA	0	0.0	0	0.0			DFW INHIBIT		0		0.0			
TOTAL	4019	11.2	0	0.0			ASYNC (TRKS)		175887		489.9			
---CKD STATISTICS---		---RECORD CACHING---		--SYNCH I/O ACTIVITY--		-HOST ADAPTER ACTIVITY-		-----DISK ACTIVITY-----						
WRITE	0	0.0	WRITE PROM	0	482919	0.000	REQ	HITS	BYTES	BYTES	RESP	BYTES	BYTES	
WRITE HITS	0	0.0					/SEC	/REQ	/REQ	/SEC	TIME	/REQ	/SEC	
							READ	0.973	4.4K	21.2M	READ	5.337	51.8K	7.2M
							WRITE	0.000	8.7K	15.0M	WRITE	13.293	89.4K	19.1M

Figure 3-45 RMF Cache Subsystem Activity report

### DASD link level (RMF ESS Synchronous I/O Link Statistics report)

The RMF ESS Synchronous I/O Link Statistics report, which is shown in Figure 3-46, shows zHyperLink from a Storage System view. It shows the link type and the I/O operations per second, bytes per operation, response time, and success %.

ESS SYNCHRONOUS I/O LINK STATISTICS													
z/OS V2R5		SYSTEM ID SY41		START 07/14/2022-16.31.00		INTERVAL 000.05.59							
SERIAL NUMBER 00000ABT11		RPT VERSION V2R5 RMF		END 07/14/2022-16.37.00		CYCLE 1.000 SECONDS							
		TYPE-MODEL 002107-986		CDATE 07/14/2022		CTIME 16.31.01		CINT 00.05.59					
SIID	-----LINK TYPE-----	---CACHE READ OPERATIONS---				---CACHE WRITE OPERATIONS---				---NVS WRITE OPERATIONS---			
		OPS	BYTES	RTIME	%SUCC	OPS	BYTES	RTIME	%SUCC	OPS	BYTES	RTIME	%SUCC
		/SEC	/OP	/OP		/SEC	/OP	/OP		/SEC	/OP	/OP	
0080	Optical PCIe GEN3 8	12657	4.1K	0.019	97.6	0.0	0.0	0.000	0.0	0.0	0.0	0.000	0.0
0081	Optical PCIe GEN1 0	NO DATA TO REPORT OR ZERO											
0180	Optical PCIe GEN3 8	12657	4.1K	0.019	97.5	0.0	0.0	0.000	0.0	0.0	0.0	0.000	0.0
0181	Optical PCIe GEN1 0	NO DATA TO REPORT OR ZERO											
0280	Optical PCIe GEN3 8	12657	4.1K	0.019	97.5	0.0	0.0	0.000	0.0	0.0	0.0	0.000	0.0
0281	Optical PCIe GEN1 0	NO DATA TO REPORT OR ZERO											
0380	Optical PCIe GEN3 8	12657	4.1K	0.019	97.5	0.0	0.0	0.000	0.0	0.0	0.0	0.000	0.0

Figure 3-46 RMF ESS Synchronous I/O Link Statistics report

### zHyperLink PFID level (RMF PCIe activity report)

Several sections of the z/OS RMF PCIe report provide useful information for monitoring zHyperLink activity:

- ▶ The General PCIe Activity section of the PCIe report, which is shown in Figure 3-47 on page 99, shows the different PCIe functions that are used, including zHyperLink (not all columns are shown).

## RMF Postprocessor Duration Report [System SY41] : PCIe Activity Report

RMF Version : z/OS V2R5 SMF Data : z/OS V2R5

Start : 07/14/2022-16.31.00 End : 07/14/2022-16.37.00 Interval : 000:06:00 hours

### ▼ General PCIe Activity

Function ID	Function CHID	Function Name	Function Status	Owner Job Name	Owner Address Space ID	Function Allocation Time	Data Transfer Rate
0254	011C	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8
0255	011C	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8
0256	011C	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8
0257	011C	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8
0264	017C	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8
0265	017C	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8
0266	017C	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8
0267	017C	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8
0274	0198	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8
0275	0198	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8
0276	0198	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8
0277	0198	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8
0284	01B8	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8
0285	01B8	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8
0286	01B8	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8
0287	01B8	8GB zHyperLink	Allocated	IOSAS	0018	360	12.8

Figure 3-47 The RMF PCIe Activity report: General PCIe Activity

- ▶ The Synchronous I/O Link Activity section of the report, which is shown in Figure 3-48, shows the detailed link activity for each zHyperLink PFID (not all columns are shown).

### ▼ Synchronous I/O Link Activity

Function ID	Function CHID	Port ID	Serial Number	Type-Model	Link ID	Total Request Rate	Total Request Rate (CPC)	Successful Request %	Successful Request % (CPC)	Read Transfer Rate	Read Transfer Rate (CPC)	Read Transfer Ratio	Read Transfer Ratio (CPC)
0254	011C	2	0000000ABT11	002107-986	0080	3184	12735	97.6	97.6	12.8	51.3	0.004	0.004
0255	011C	2	0000000ABT11	002107-986	0080	3184	12735	97.6	97.6	12.8	51.3	0.004	0.004
0256	011C	2	0000000ABT11	002107-986	0080	3184	12735	97.5	97.6	12.8	51.3	0.004	0.004
0257	011C	2	0000000ABT11	002107-986	0080	3184	12735	97.6	97.6	12.8	51.3	0.004	0.004
0264	017C	2	0000000ABT11	002107-986	0180	3184	12735	97.5	97.5	12.8	51.3	0.004	0.004
0265	017C	2	0000000ABT11	002107-986	0180	3184	12735	97.5	97.5	12.8	51.3	0.004	0.004
0266	017C	2	0000000ABT11	002107-986	0180	3184	12735	97.5	97.5	12.8	51.3	0.004	0.004
0267	017C	2	0000000ABT11	002107-986	0180	3184	12735	97.5	97.5	12.8	51.3	0.004	0.004
0274	0198	2	0000000ABT11	002107-986	0280	3184	12735	97.5	97.5	12.8	51.3	0.004	0.004
0275	0198	2	0000000ABT11	002107-986	0280	3184	12735	97.6	97.5	12.8	51.3	0.004	0.004
0276	0198	2	0000000ABT11	002107-986	0280	3184	12735	97.5	97.5	12.8	51.3	0.004	0.004
0277	0198	2	0000000ABT11	002107-986	0280	3184	12735	97.5	97.5	12.8	51.3	0.004	0.004
0284	01B8	2	0000000ABT11	002107-986	0380	3184	12735	97.5	97.5	12.8	51.3	0.004	0.004
0285	01B8	2	0000000ABT11	002107-986	0380	3184	12735	97.6	97.5	12.8	51.3	0.004	0.004
0286	01B8	2	0000000ABT11	002107-986	0380	3184	12735	97.5	97.5	12.8	51.3	0.004	0.004
0287	01B8	2	0000000ABT11	002107-986	0380	3184	12735	97.5	97.5	12.8	51.3	0.004	0.004

Figure 3-48 The RMF PCIe: Synchronous I/O Link Activity



- ▶ The Synchronous I/O Response Time Distribution report, which is shown in Figure 3-49, shows the distribution of the read/write response times in milliseconds for each of the synchronous I/O PFIDs (not all columns are shown).

▼ Synchronous I/O Response Time Distribution										
Function ID	% Read < 20usec	% Read < 30usec	% Read < 40usec	% Read < 50usec	% Read < 60usec	% Read < 70usec	% Read < 80usec	% Read < 90usec	% Read < 100usec	% Read >=100usec
0254	4.33	95.0	0.347	0.282	0.053	0.005	<.001	<.001	<.001	<.001
0255	4.37	94.9	0.344	0.293	0.054	0.005	<.001	<.001	0	<.001
0256	4.32	95.0	0.355	0.284	0.056	0.006	<.001	<.001	<.001	<.001
0257	4.34	95.0	0.345	0.274	0.060	0.006	<.001	<.001	<.001	<.001
0264	8.15	91.1	0.376	0.278	0.052	0.006	0.002	<.001	<.001	<.001
0265	8.18	91.1	0.377	0.269	0.050	0.008	<.001	<.001	<.001	<.001
0266	8.21	91.1	0.382	0.275	0.052	0.006	<.001	<.001	<.001	<.001
0267	8.26	91.0	0.387	0.283	0.053	0.006	0.001	<.001	<.001	<.001
0274	2.30	97.0	0.358	0.288	0.058	0.005	<.001	<.001	0	<.001
0275	2.31	97.0	0.355	0.296	0.057	0.006	<.001	<.001	0	0
0276	2.30	97.0	0.356	0.281	0.061	0.005	<.001	<.001	0	0
0277	2.31	97.0	0.350	0.293	0.058	0.005	<.001	0	0	0
0284	4.18	95.1	0.393	0.288	0.057	0.006	<.001	<.001	<.001	<.001
0285	4.18	95.1	0.376	0.284	0.057	0.005	0.001	<.001	<.001	0
0286	4.19	95.1	0.385	0.286	0.058	0.006	<.001	<.001	<.001	<.001
0287	4.20	95.1	0.376	0.287	0.054	0.006	<.001	<.001	0	<.001

Figure 3-49 The RMF PCIe: Synchronous I/O Response Time Distribution

**Storage class and data set level SMF records**

Storage class and data set level SMF records (type 42-5 and 6) also include zHyperLink information. For more information, see the following topics:

- ▶ [Synchronous I/O section 1](#)
- ▶ [Synchronous I/O section 2](#)
- ▶ [Synchronous I/O section 3](#)

**3.10.6 zHyperLink read usage considerations**

Db2 11 and later can leverage zHyperLink for database synchronous reads. However, before you implement zHyperLink, you should be aware of some important considerations:

- ▶ Figure 3-50 on page 101 shows the different locations where Db2 looks for the data that it needs to process SQL requests, with the fastest location at the top and the slowest at the bottom. When Db2 accesses the data, the fastest method is to access the data that is in memory. A good example is the index in-memory (or [Fast Traversal Blocks \(FTB\)](#)) capability that was introduced in Db2 12. If the pages are not found in such an in-memory area outside of buffer pools, the next best place to find them is in the Db2 buffer pools to avoid I/O operation. The next best option is zHyperLink technology, which can be faster than the standard I/O from the disk cache. If the data is not in the disk cache, a disk access (SSD or HDD) must be performed.

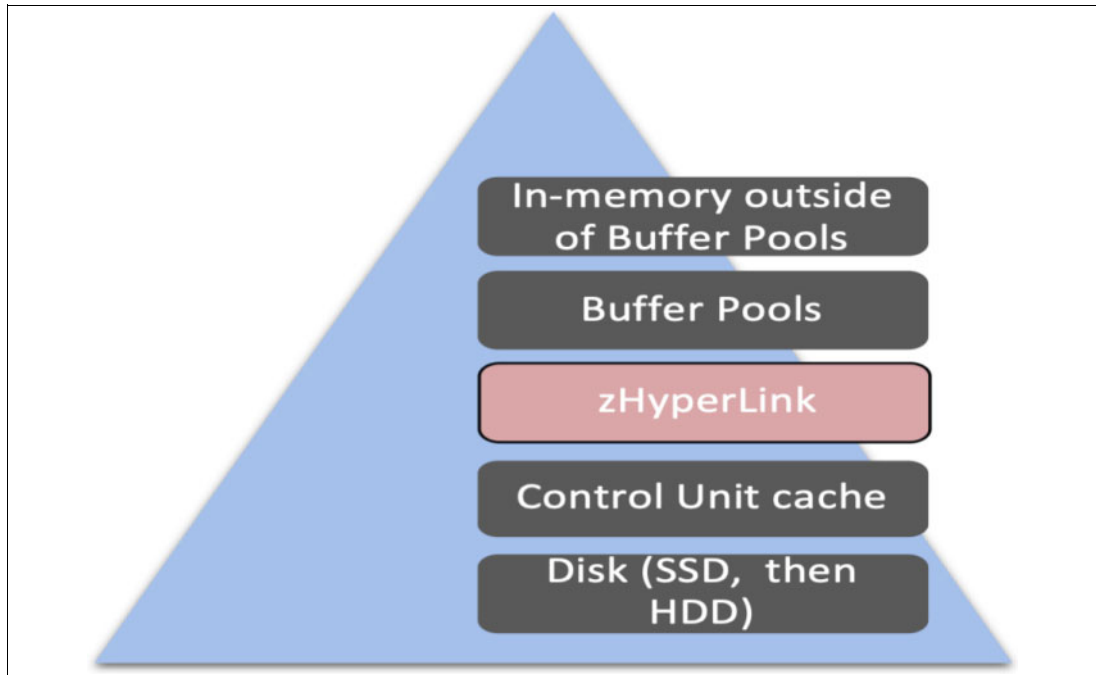


Figure 3-50 Db2 database read access

The solution that provides the greatest performance improvement is to invest in larger memory to optimize the performance. However, zHyperLink read support can help if you need to improve or scale I/O-bound transactions without increasing your memory footprint.

Only the data that is found in the disk control unit cache is eligible for zHyperLink read I/Os. Therefore, workloads with many synchronous random reads that are cache hits in the DASD subsystem are generally good candidates for using zHyperLink read. Such workloads show substantial transaction elapsed time improvements because of the dramatic reduction in read I/O latency. Enabling zHyperLink if the disk cache hit rate is less than 75 - 80% provides little to no performance benefit.

- ▶ zHyperLink limits database random read I/O to a 4 K Control Interval (CI). Therefore, 8 K, 16 K, and 32 K CIs are not eligible for zHyperLink read I/O. The zHyperLink I/O-eligible reads might include I/Os from a buffer pool other than 4 K. For example, when a compressed index uses a buffer pool page size greater than 4 K, the page on disk for this index data set is always 4 K; therefore, the CI size is 4 K. Db2 reads compressed index pages from disk as 4 K pages and expands them to the buffer pool page size.
- ▶ The large reduction in elapsed time when low-latency zHyperLink I/O is used comes with a CPU cost. Although the zHyperLink I/O latency is a fixed clock time, the CPU cost varies depending on the processor speed. The faster the speed of the processor, the more CPU cycle spins during the fixed clock time. The negative CPU impact is less on a slower processor.

In a busy system in which CPU utilization is high (over 85%), the CPU cost of processing traditional asynchronous I/O might be high. In traditional I/Os, the CPU cost of the interrupt handling process and dispatching delay, along with the associated Level1/Level2 processor cache misses, increases the code path length. zHyperLink I/O latency is low and it eliminates interrupt handling process and dispatching delay. Therefore, in a busy system, the CPU impact of zHyperLink might not be significant when it is compared to traditional asynchronous I/O.

### 3.10.7 zHyperLink read conclusion

The measurements that we obtained show that a workload with many random reads that are eligible for zHyperLink can achieve dramatic transaction response time improvements.

The shorter transaction response time results in an increase in CPU cost for high-end IBM zSystems. However, the CPU usage increase for the target workload is a reasonable tradeoff for the tremendous reduction in transaction elapsed time when 95% of the database reads are cache hits and the LPAR CPU is busy with other background work.

For mid-range systems, such as IBM z14 ZR1, which has subcapacity processors, a reduction in transaction elapsed time can be attained with minimal to even a positive effect on CPU usage. zHyperLink I/O is a good option for database random read I/O for mid-range systems such as ZR1 in which memory expansion is limited and increasing buffer pool size might not be an option.

The low latency of zHyperLink I/O makes it possible to achieve improved transaction response time with no changes to user applications.

### 3.10.8 zHyperLink and writes

Active log force write I/O can impact transaction commit time. This impact can be profound for transactions with heavy update processing, especially when index splits occur in a data-sharing environment. The low I/O latency of zHyperLink technology is leveraged by Db2 for active log force writes, which reduces the I/O latency time of active log write, speeds up commit time, and improves transaction elapsed time. Unlike zHyperLink read support for database random read I/O that is available in Db2 11 and later, zHyperLink write support is available only for Db2 12 at FL 100 and later.

In 2019, zHyperLink write was introduced only for simplex and synchronous copy (Metro Mirror).

In 2020, for z/OS 2.3 and later, zHyperLink write support was extended to asynchronous copy (Global Mirroring but not for Extended Remote Copy (XRC)). For more information about the zHyperLink write capability for synchronous and asynchronous mirroring, see [zHyperLink Write Support](#).

Initially, when Db2 leveraged zHyperLink for active log writes, it did not run parallel writes in a dual active log configuration. In 2021, z/OS DFSMS Media Manager added support to Db2 to write to dual active logs in parallel with a single interface call.

The Db2 writes to active logs are processed under an enclave SRB that runs in the Db2 system services (*ssnmMSTR*) address space. The CPU cost of spinning while waiting for the active log write I/O to complete is 100% zIIP eligible and does not add to operational cost. The reduced log write latency with no extra operational cost and no changes to user applications makes zHyperLink Write I/O a viable feature to improve Db2 transaction performance.

### 3.10.9 Requirements

The following environment is required to leverage zHyperLink write for active logs:

- ▶ IBM z14 or later
- ▶ IBM DS8880 or later storage subsystem
- ▶ z/OS 2.3 or later
- ▶ z/OS Media Manager APARs OA57833, OA58134, and OA59581
- ▶ Db2 12 (with APAR PH29407) at FL 100 or later
- ▶ The PTFs for zHyperLink, which you can find by searching for Fix Category `IBM.Function.zHyperLink` with APAR keyword `HYPERL/K`

### 3.10.10 Performance measurements

A number of performance measurements were run with zHyperLink enabled for active log writes. We used an insert-intensive workload and the IBM brokerage transaction workload to demonstrate the reduction in elapsed time and the effect of using zHyperLink technology on CPU usage.

#### Measurement environment

Performance measurements for using zHyperLink write I/O for active log writes are run in the following environments:

- ▶ Db2 12 at FL 508, running on z/OS 2.4 with necessary maintenance.
- ▶ Db2 13 at FL 501, running on z/OS 2.5 with necessary maintenance.
- ▶ IBM z14 M05-3907-7E7 connected to an IBM DS8886 system.
- ▶ IBM z15 connected to an IBM DS8886 system.
- ▶ **INSERT** Intensive Workload on Db2 12 FL 508 and IBM z14.
- ▶ IBM brokerage transaction workload on Db2 FL501 and IBM z15.
- ▶ All measurements do not use striped active logs unless otherwise described in the test scenario.

#### Measurement scenario description

The following measurement scenarios were run to determine the types of workloads that would benefit from enabling zHyperLink for active log writes and the effect of using zHyperLink on CPU consumption:

- ▶ An insert-intensive workload in a one-way data-sharing environment that uses zHyperLink and zHPF link for active log writes.
- ▶ An insert-intensive workload in two-way data-sharing environment that uses zHyperLink and zHPF link for active log writes.
- ▶ An insert-intensive workload in one-way data-sharing environment that uses zHyperLink and zHPF link for active log writes with a log stripe size of 4.
- ▶ IBM brokerage transaction workload with different options that are specified for the Db2 **ZHYPERLINK** subsystem parameter.

#### Results

Workloads that are update-intensive with frequent commits benefit the most from implementing zHyperLink for Db2 active log writes.

Figure 3-51 shows an insert-intensive workload in a one-way data-sharing environment with 50 threads concurrently inserting 10 rows per commit to different partitions in a table.

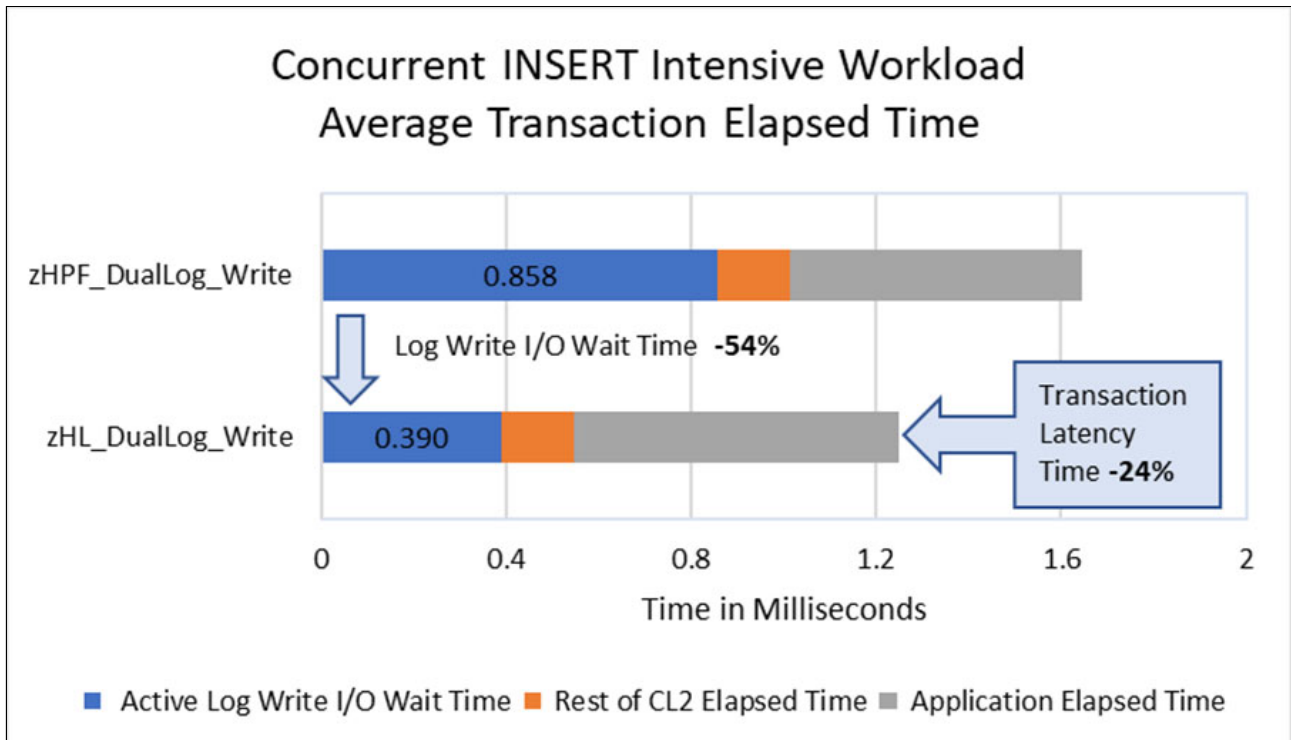


Figure 3-51 zHyperLink write benefit with an insert-intensive workload

Using zHyperLink I/O to write to dual active logs results in a 54% reduction in class 3 active log write I/O wait time in Db2. This reduction improved transaction latency by 24% compared to active log writes that use traditional I/O through zHPF links.

In a data-sharing environment, index splits of GBP-dependent indexes cause forced log writes. To maintain data integrity, a Db2 latch is held on the index page set while data is written to the active log data set. Db2 latch contention can be high in such situations, and they often become the bottleneck in an insert-intensive workload. The low latency of zHyperLink I/O can reduce Db2 latch contention and improve transaction throughput.

To demonstrate this scenario, measurements were run with an insert-intensive workload in a two-way data-sharing environment with 100 threads inserting concurrently 10 rows per commit into different partitions. The table was set up with three NPIs to cause index splits and to force log writes. The results are shown in Figure 3-52 on page 105.



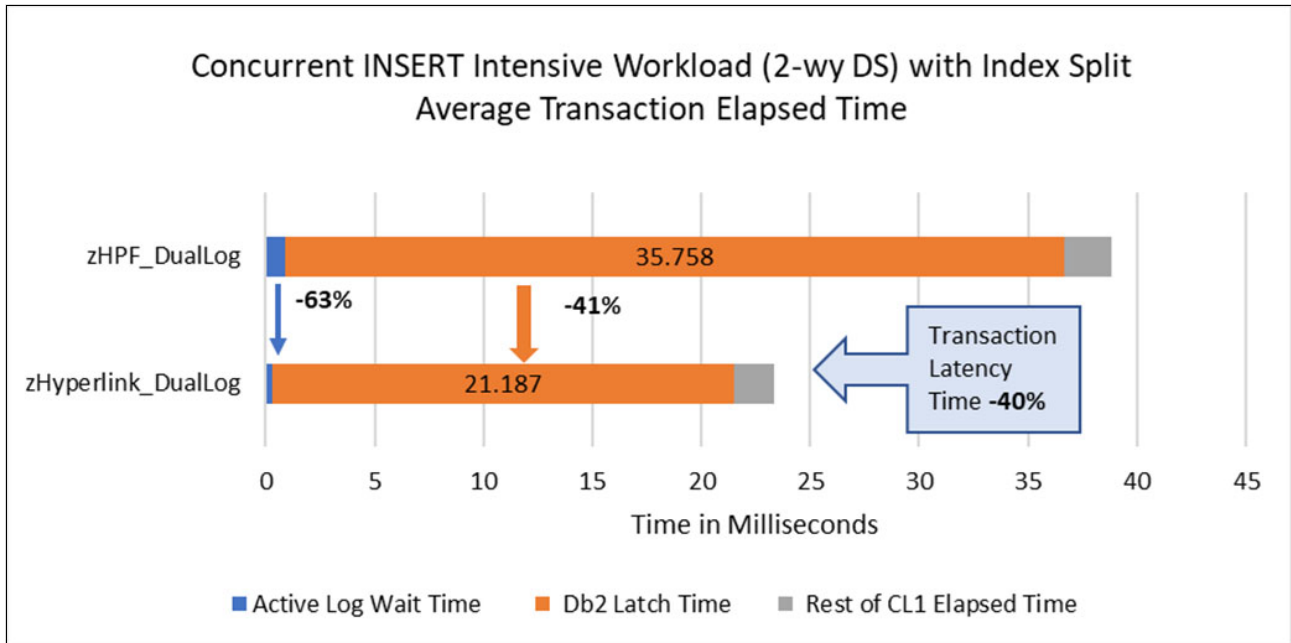


Figure 3-52 zHyperLink write benefit with heavy index splits

When zHyperLink write was used for dual active logs, we observed a 63% reduction in log write I/O wait time in Db2, and then a 41% reduction in Db2 latch time. The drastically reduced active log write zHyperLink I/O time and fewer Db2 latch contentions improved transaction latency time by 40% when compared to using zHPF for writing to dual active logs.

The IOS component of z/OS limits zHyperLink writes to two tracks. Db2 active log writes can be up to 512 K (128 4-K records) or up to 12 tracks. Therefore, when the log write size gets larger, more I/O requests occur when zHyperLink is used. Because the I/O latency is drastically reduced with zHyperLink, the average transaction latency is still reduced compared to using a zHPF link.

The next set of measurements show a reduction in transaction response time despite a large log write size when zHyperLink is used. The same insert-intensive workload was run after enabling zHyperLink write for active log data sets with a stripe size of 4 on uncompressed data to increase the log write size to 128 4-K records. We increased the log write size by increasing the number of inserts per commit.

The results are summarized in Figure 3-53.

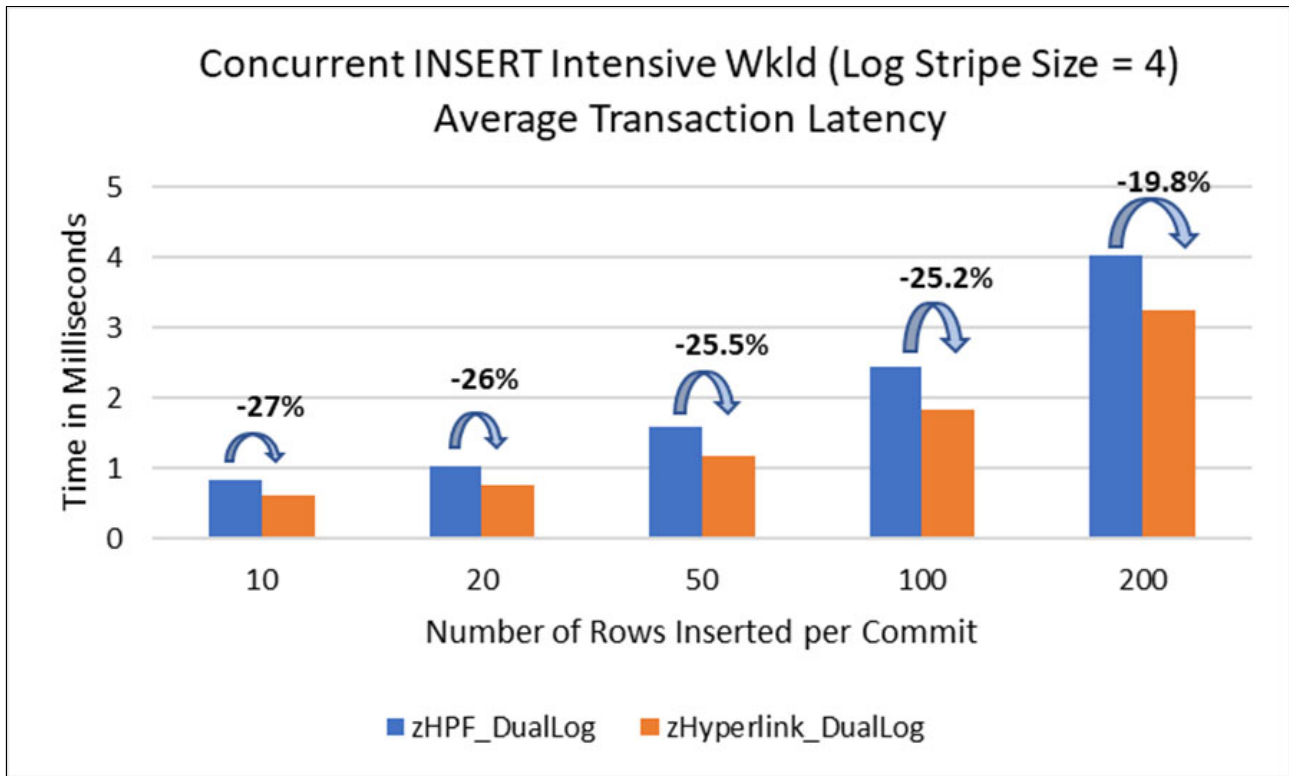


Figure 3-53 The average latency with different commit frequency

We observed that as the log write size increased, the transaction latency time improvement reduced. The cause of this behavior is that when zHyperLink is used for active log writes, there is a two-track limit per I/O when writing in parallel to dual logs. This limitation increases the number of log write I/O requests compared to using a zHPF link, as shown in Figure 3-54. However, we still observed an approximately 20% reduction in the average response time.

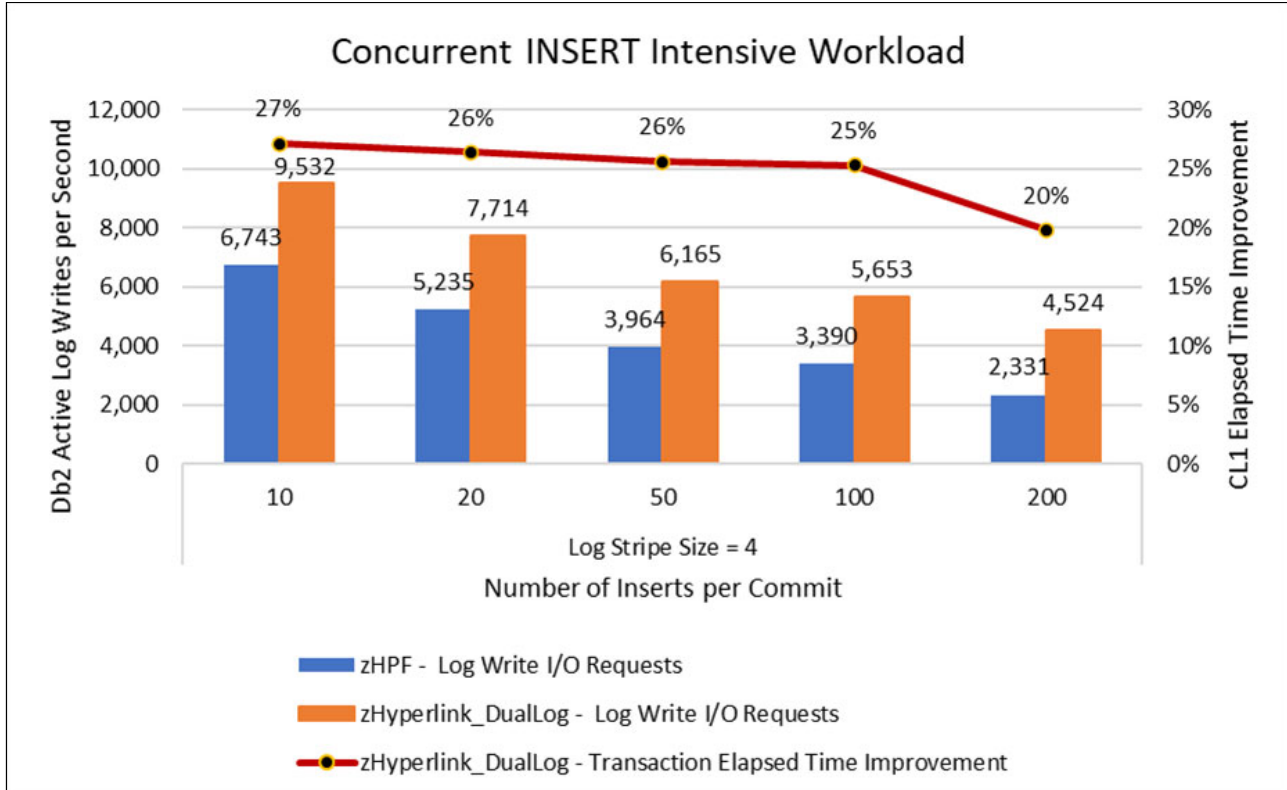


Figure 3-54 The active log writes with different commit frequency

Using zHyperLink for active log writes also results in higher CPU usage because the CPU spins until the I/O operation completes. However, unlike zHyperLink database random read I/O, active log write is charged to the Db2 System Services (MSTR) address space CPU under an SRB, and this work is zIIP eligible. zIIP usage does not incur extra charges.

For our insert-intensive workload that used zHyperLink writes for Db2 active logs, we observed a substantial increase in zIIP CPU utilization with no increase in class 2 CPU time, as shown in Figure 3-55. Therefore, you must consider your current usage of zIIP processors and plan for more zIIP capacity as needed before implementing zHyperLink writes for active logs.

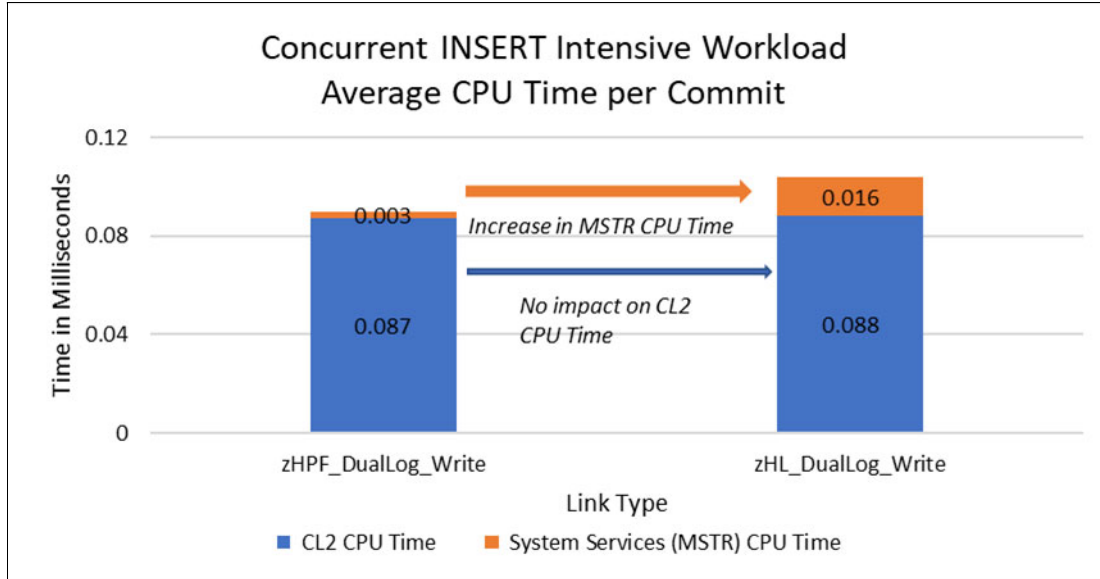


Figure 3-55 CPU usage per commit with zHyperLink write

Workloads with a high read I/O percentage might not always show improvement in transaction rates if the active log write I/O time is not a major factor that is impacting transaction elapsed time. The next set of measurements demonstrate that for these workloads that zHyperLink writes did not result in a noticeable transaction elapsed time improvement, as shown in Figure 3-56.

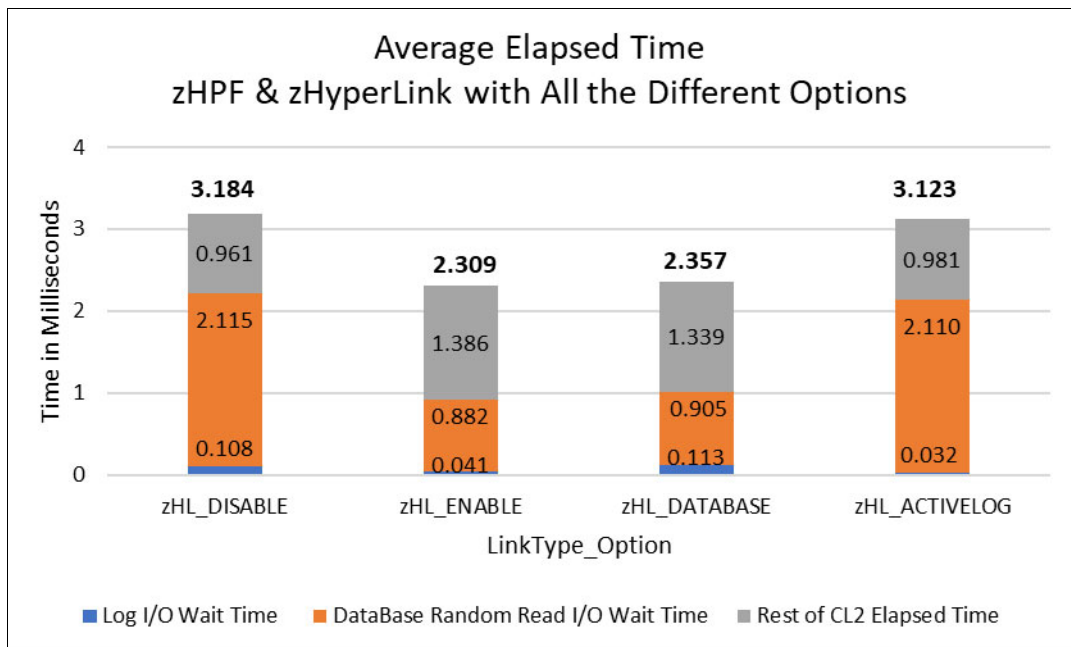


Figure 3-56 Response time with different Db2 ZHYPERLINK options

For the IBM brokerage transaction workload, the write to active log data set is not the major bottleneck to achieving improved throughput, as shown in the zHL\_DISABLE bar. Here, most of the transaction delay is from database random read I/O wait time.

The results from these measurements demonstrate the benefit of different zHyperLink enablement options (that is, different **ZHYPERLINK** subsystem parameter settings) on improving transaction elapsed time.

When zHyperLink is enabled for active log write I/O only with the **ACTIVELOG** option, we observed an approximately 1.9% reduction in elapsed time compared to using zHPF for log write I/O.

When zHyperLink is enabled for database random read I/O with the **DATABASE** option, elapsed time is substantially reduced by 25.9% compared to using zHPF for read I/O.

When zHyperLink is enabled for both active log write I/O and database random read I/O by using the **ENABLE** option, we observed a further reduction of 27.4% in elapsed time compared to using zHPF for both reads and writes.

### Summary of results of using zHyperLink write

Our results show that the transaction latency time can be drastically reduced for update-intensive workloads when zHyperLink is used for active log write I/O compared to using zHPF. The log wait time was reduced by 54%, and then the elapsed time was reduced by 24%. In an update-intensive workload with index splits in a two-way data-sharing environment, the log wait time was reduced by 69%, which further reduced Db2 latch time and improved elapsed time by 40%.

IBM also conducted a performance study that involved the SAP Core Banking Account Settlement workload with Db2 12 that used zHyperLink link for active log write I/O. This workload simulates account balancing by calculating the interest and charges of all the accounts in the test database. The Db2 12 database that was used in this study contains a total of 100 million banking accounts, which is comparable to some of the largest banks in the world. The result of this evaluation also shows 20% reduction in total batch elapsed time when zHyperLink is used for active log write I/O compared to when zHPF link is used for active log write I/O. For more information, see [Evaluation of zHyperLink Write for Db2 Active Logging with SAP Banking on IBM Z](#).

During our striped active log data sets measurements, we increased the log write size by increasing the number of inserts per commit. We observed that the elapsed improvement reduced as we increased the log write size by inserting more records per commit. We still observed a minimum of 20% elapsed time improvement.

A CPU increase for the Db2 *ssnm*MSTR address space occurs when zHyperLink is used for active log writes. This CPU time is fully zIIP eligible, which does not add to operational cost. No increase in the transaction class 2 CPU time was observed.

The evaluation of the different options of the **ZHYPERLINK** subsystem parameter with the IBM brokerage transaction workload illustrates that not all workloads benefit equally from using zHyperLink. Therefore, before setting **ZHYPERLINK**, determine where the bottlenecks to achieving lower transaction latency are in your environment.

## 3.10.11 Monitoring zHyperLink write

Performance information that is related to zHyperLink write I/O is collected by Db2 and z/OS.

## Db2 monitoring of zHyperLink active log write I/O

Db2 added several new fields to the statistics record to monitor zHyperLink I/O:

► SMF100 statistics record

The IFCID 1 trace record (Log Manager section) was updated with the following fields to monitor active log writes that use zHyperLink:

- QJSTSYCW contains the total number of eligible Log Write I/O requests with successful zHyperLink processing.
- QJSTSYCF contains the total number of eligible Log Write I/O requests with unsuccessful zHyperLink processing.

► DISPLAY command

You can use the **-DISPLAY LOG** command to determine whether active log data sets are using zHyperLink (write) I/O, as shown in Figure 3-57.

```
-CEA2 DIS LOG
DSNJ378I -CEA2 DSNJC00A LOG DISPLAY 894
CURRENT COPY1 LOG = DB2CEA0.CEA2.LOGCOPY1.DS08 IS 56% FULL
CURRENT COPY2 LOG = DB2CEA0.CEA2.LOGCOPY2.DS08 IS 56% FULL
      H/W RBA = 0000000000493C59DA58
      H/O RBA = 00000000000000000000
      FULL LOGS TO OFFLOAD = 0 OF 8
      OFFLOAD TASK IS (AVAILABLE)
      SOFTWARE ACCELERATION IS DISABLED
      ZHYPERLINK WRITE IS ENABLED
      LAST LOG WRITE FOR COPY1 USED ZHYPERLINK
      LAST LOG WRITE FOR COPY2 USED ZHYPERLINK
DSNJ371I -CEA2 DB2 RESTARTED 15:38:46 JUL 13, 2022 895
      RESTART RBA 000000000040B6C4F000
      CHECKPOINT FREQUENCY 10000000 LOGRECORDS
      LAST SYSTEM CHECKPOINT TAKEN 16:17:30 JUL 13, 2022
DSN9022I -CEA2 DSNJC001 '-DIS LOG' NORMAL COMPLETION
```

Figure 3-57 DISPLAY LOG command output with zHyperLink

## z/OS monitoring of zHyperLink write I/O information

z/OS also monitors zHyperLink I/O activity, including write activity, at different levels. You can use the following reports and command to gain better insights:

- The RMF reports that are described in 3.10.5, “Monitoring Db2 zHyperLink database I/O” on page 95.
- Storage class and data set level SMF records, type 42-5 and 42-6, also include zHyperLink information. For more information, see the following topics:
  - [Synchronous I/O section 1](#)
  - [Synchronous I/O section 2](#)
  - [Synchronous I/O section 3](#)
- The z/OS DFSMS Media Manager display command:  
D SMS,DSNAME(dsn),STATS(ZHLWRITE{,RESET})

For more information about the command output, see [IGW289I D SMS,DSNAME,STATS\(ZHLWRITE\) Start of Report](#).

Figure 3-58 shows the **DISPLAY** command and the output that can be used to display the percentage of synchronous I/O being done.

```

IGW289I 662
D SMS,DSNAME,STATS(ZHLWRITE) Start of Report
DATA SET DB2C1NO.DC1N.LOGCOPY1.DS01.DATA
STATISTICS Since 02/05/2021 07:38:15.373500
SUMMARY
TOTAL          %SYNC  -----%ASYNC-----
WRITE REQUESTS WRITES  SKIP LNKBSY  ^EST  MISC DISABL
51346  99.99    0.00  0.00  <0.01  0.00  0.00
-----%ASYNC-----
MISC DELAY DUAL
0.00  0.00  0.00
DEVICE STATISTICS
TOTAL          %SYNC  -----%ASYNC-----
SSID DEVNO WRITES WRITES  SKIP LNKBSY  ^EST  MISC MISS DELAY
AD00 0B024 51826 99.99    0.00  0.00  <0.01  0.00  0.00  0.00
D SMS,DSNAME,STATS(ZHLWRITE) End of Report

```

Figure 3-58 DFSMS Media Manager showing zHyperLink activity

### 3.10.12 zHyperLink write usage considerations

When commit processing time is crucial for performance, zHyperLink provides tremendous elapsed time improvements without modifying your applications. The active log force writes during an index tree modification against GBP-dependent indexes is a good example. The tree modification can be an index leaf page split from an insert operation or an index page consolidation from a delete activity. During the tree modification, Db2 holds an internal latch against the index tree while it writes to active log data sets. This operation is essential to maintaining the data integrity. However, it often causes a bottleneck in insert- or delete-intensive applications in a data-sharing environment. The low latency from zHyperLink I/O can reduce the duration of the active log write and Db2 latch wait, which improves overall Db2 application throughput.

When dual active logging is used, both log data sets must be on zHyperLink enabled DASD to be eligible for zHyperLink write.

zHyperLink writes can write up to two tracks per I/O request due to a z/OS IOS limitation. This limitation can result in more log I/Os for the same workloads when zHyperLink for active log is enabled. zHyperLink write support for synchronous mirroring is available on the IBM DS8880 and IBM DS8900. For Metro Mirror environments, the devices on which the Db2 log data sets are on must be in the duplex state and in a HyperSwap configuration, and zHyperWrite must be enabled. All IBM DS8800 devices in a Metro Mirror environment must be within 150 meters, and all of them must be connected through zHyperLink. If Metro Mirror (PPRC) is enabled for the active log data sets, zHyperLink uses zHyperWrite technology to write to the secondary. When you enable **ZHYPERLINK** write, z/OS enables zHyperWrite for peer write regardless of the value that you specify for the zHyperWrite subsystem parameter (**REMOTE\_COPY\_SW\_ACCEL**) if the distance between primary and secondary is within 150 meters.

zHyperLink write support for asynchronous mirroring is available only on the DS8900.

CPU spin time while waiting for zHyperLink active log write completion is accounted for in System Services (MSTR) SRB time, which is zIIP eligible. Therefore, you must plan for extra zIIP capacity before you enable zHyperLink for active log write I/O.

### 3.10.13 zHyperLink write conclusion

Our results demonstrate that when commit processing is a bottleneck for good performance, low-latency zHyperLink I/O for active log writes can improve transaction elapsed time. This performance benefit can be achieved without modifying your applications.

The CPU increase for the Db2 *ssnm*MSTR address space is fully zIIP eligible with no increase in transaction class 2 CPU time. You must plan for more zIIP capacity before implementing zHyperLink writes for Db2.

When implementing zHyperLink for Db2, evaluate your workloads to assess zHyperLink eligibility and benefits. zBNA is a useful tool for evaluating the eligibility and benefits of leveraging zHyperLink I/O.

## 3.11 Huffman compression for data

Db2 introduced table space and partition-level compression in Version 3. The algorithm that is used for table space compression is based on a Ziv tree. When this fixed-length compression algorithm is used, a hardware instruction that is called CMPSC replaces a repeating sequence in the Ziv tree with a 12-bit fixed-length symbol.

Huffman compression, also referred to as *entropy encoding*, uses the same Ziv-tree-based mechanism, but it accounts for the sequence frequency. It uses variable-length codes to replace sequences. More frequent sequences are replaced with shorter symbols. This replacement is done by adding a symbol translation stage, during which the fixed-length node index is mapped into a variable-length symbol lengthening from 1 - 16 bits. By using such a methodology, more data can be replaced with shorter symbols, which achieves a better compression ratio.

To achieve optimal performance, an enhanced CMPSC hardware instruction was implemented on IBM z14 and later. For this reason, compressing Db2 data by using Huffman compression can be done only on IBM z14 and later. However, a new Expansion Preprocessor Routine (EPROC) enables you to decompress data that was compressed by using Huffman compression on older hardware that does not have the enhanced CMPSC instruction. However, the performance cost is high, so be cautious and try to avoid such a situation.

When compared with fixed-length compression, Huffman compression provides a higher compression ratio, which means that it saves more space and reduces the cost of managing data sets. If maintaining your data volume is expensive and requires an inordinate amount of effort, you might want to consider using Huffman compression.

A better compression ratio also means that the same number of records can be contained with fewer data pages, which result in less getpage activity during query execution. This reduction saves elapsed time, especially when large amounts of data, typically brought into the buffer pool through sequential prefetch, are involved. If you want to reduce the run time of your queries, you can choose Huffman compression for these objects to reduce the number of getpages that must be processed.

### 3.11.1 Enabling Huffman compression

This section describes the requirements for enabling Huffman compression and provides some information about how you can use Huffman compression.

#### **Hardware and software requirements for compression**

To use Huffman compression, your environment must meet the following requirements:

- ▶ IBM z14 processor or later
- ▶ Db2 12 at FL 504 or later



- ▶ The Db2 `TS_COMPRESSION_TYPE` subsystem parameter set to `HUFFMAN`.  
This subsystem parameter also determines the type of compression that is used for the directory table space (SPT01).
- ▶ The universal table space (UTS) `COMPRESS` attribute is set to `YES`.

When all the requirements are satisfied, Huffman compression is enabled.

Starting at Db2 12 FL 509 you also can specify the usage of Huffman compression at the individual table space or partition level by using the `COMPRESS YES HUFFMAN` option.

If any of the prerequisites are not met, a classic dictionary is created, and data is compressed by using the classic (fixed-length) algorithm. If a Huffman dictionary exists for the table space or table space partition, the record is stored uncompressed.

### Requirements for decompressing Huffman compressed data

To decompress data that was compressed by using Huffman compression, you need Db2 12 at FL 504 or later.

Although there is no specific hardware requirement for decompressing Huffman-compressed data, doing this task on a machine older than IBM z14 is not recommended because the software decompression uses much CPU time.

### Using Huffman compression

Data rows can be compressed during insert operations or when running a `LOAD` or `REORG` utility.

#### *Compression on insert*

When the requirements are met, a Huffman dictionary can be built by inserting data into a table space or partition. After the inserted data volume exceeds a threshold (which is determined by Db2), a Huffman dictionary is built, and when subsequent data is inserted (or updated), it is compressed by using this dictionary.

Starting at Db2 12 FL 509, Huffman compression can be specified at the table space or table part level. If there is existing data in a table space that is compressed with the fixed-length algorithm (that is, a fixed-length dictionary exists) and you alter the table space to `COMPRESS YES HUFFMAN`, subsequent inserts continue to use the fixed-length dictionary to compress rows until you do a `REORG` to convert the dictionary to a Huffman dictionary.

#### *Compression by LOAD*

Huffman compression can also be used with the `LOAD` utility. When you load data into an empty table space or partition without specifying `KEEPDICTIONARY`, or when you load data with `REPLACE` without `KEEPDICTIONARY`, Db2 builds a compression dictionary while the records are loaded. After the dictionary is built, subsequent loaded data is compressed by using this dictionary.

#### *Compression by REORG*

The `REORG` utility can be used to convert a table space or partition that was compressed by using the classic fixed-length algorithm to Huffman compression, or to convert uncompressed data to Huffman compressed data. To do so, ensure that the Huffman compression prerequisites are met, and `REORG` the table space or partition without specifying `KEEPDICTIONARY`.

The **REORG** utility typically provides a better compression ratio than **INSERT** and **LOAD** because it samples more records to build the dictionary. For more information, see 3.11.2, “Space savings from Huffman compression” on page 114.

### ***Evaluating the compression ratio***

To determine the actual compression ratio and the DASD savings, check the compression report that is produced by the **LOAD** and **REORG** utilities. Messages DSNU234I and DSNU244I contain the data size, row length, and number of pages before and after compression.

Information about compression savings is also available in Db2 catalog tables if the **RUNSTATS** utility or inline statistics were run against the table space. The **PAGESAVE** column in **SYSIBM.SYSTABLEPART** contains the percentage of pages that are saved.

You can also use the **DSN1COMP** stand-alone utility to estimate the amount of compression that you can expect without compressing the table space or partition. For more information, see 3.11.5, “Using DSN1COMP for Huffman compression estimation” on page 125.

## **3.11.2 Space savings from Huffman compression**

A number of performance measurements were conducted to evaluate the Huffman compression feature. The compression ratio and the performance of compression and decompression were analyzed.

### **Measurement environment**

The following system configuration was used to conduct these measurements:

- ▶ IBM z14 (software decompression on IBM z13) with six general CPs, one zIIP, and 64 GB of memory
- ▶ z/OS 2.2
- ▶ Stand-alone Db2 12 at FL 504

### **Compression ratio**

Because Huffman compression uses a more efficient algorithm, the compression ratio also is expected to improve when compared to using fixed-length compression. The **LOAD** and **REORG** utilities were used to evaluate the compression ratio of Huffman compression against the set of workloads.

### ***Compression ratio when using the LOAD utility***

When loading data into the table, we observed a compression ratio of 40% - 95% with an average of 70%. Figure 3-59 on page 115 shows the compressed data volume in which the **LOAD** utility was used with the three separate workloads.

- ▶ Depending on the data, Huffman compression shows an average improvement of 10.5% when compared to fixed-length compression. The improvement can be up to 44.6% in some cases. **DSN1COMP** can be used to estimate the efficiency of Huffman compression to facilitate decision making.
- ▶ Depending on the data, Huffman compression can reduce DASD usage by an extra 3.3% on average when compared to fixed-length compression. This number can be up to 14.7% in some circumstances.

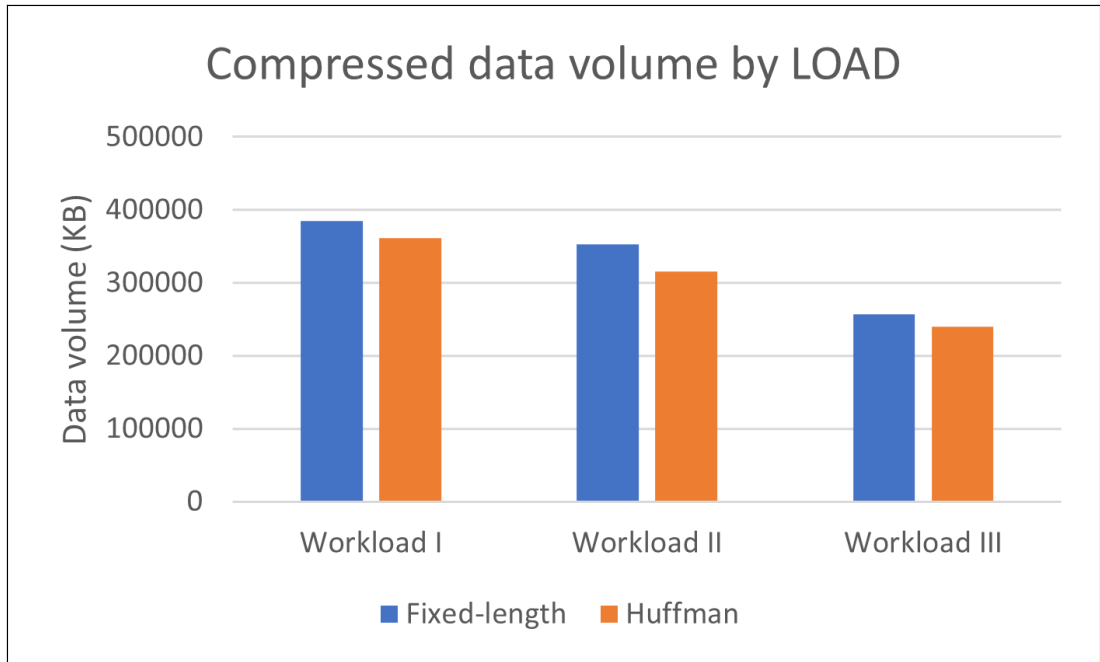


Figure 3-59 Compressed data size by using the LOAD utility

### **Compression ratio when using the REORG utility**

Unlike fixed-length compression, which has a similar compression ratio for both **LOAD** and **REORG**, the Huffman compression ratio derives better performance from more data sampling. The **REORG** utility doubles the sampling data compared to the **LOAD** utility when building the dictionary, which enables it to achieve a better compression ratio. Therefore, **REORG** is recommended if you want to save more space by using Huffman compression.

During our measurements, Huffman compression achieved a 60%- 95% compression ratio with an average of 80%.

Figure 3-60 shows the compressed data volume in which the REORG utility was used with three separate workloads:

- ▶ Compared to fixed-length compression, using Huffman compression with the **REORG** utility provides an average improvement of 31.3%. The improvement can be up to 57.9% in some cases. How efficient Huffman compression is depends on the distribution of the characters. **DSN1COMP** can be used to estimate the efficiency of Huffman compression to facilitate decision making.
- ▶ Compared to fixed-length compression, using Huffman compression with the **REORG** utility can reduce the DASD usage by another 9.3% on average. This number can be up to 28.4% in some circumstances.

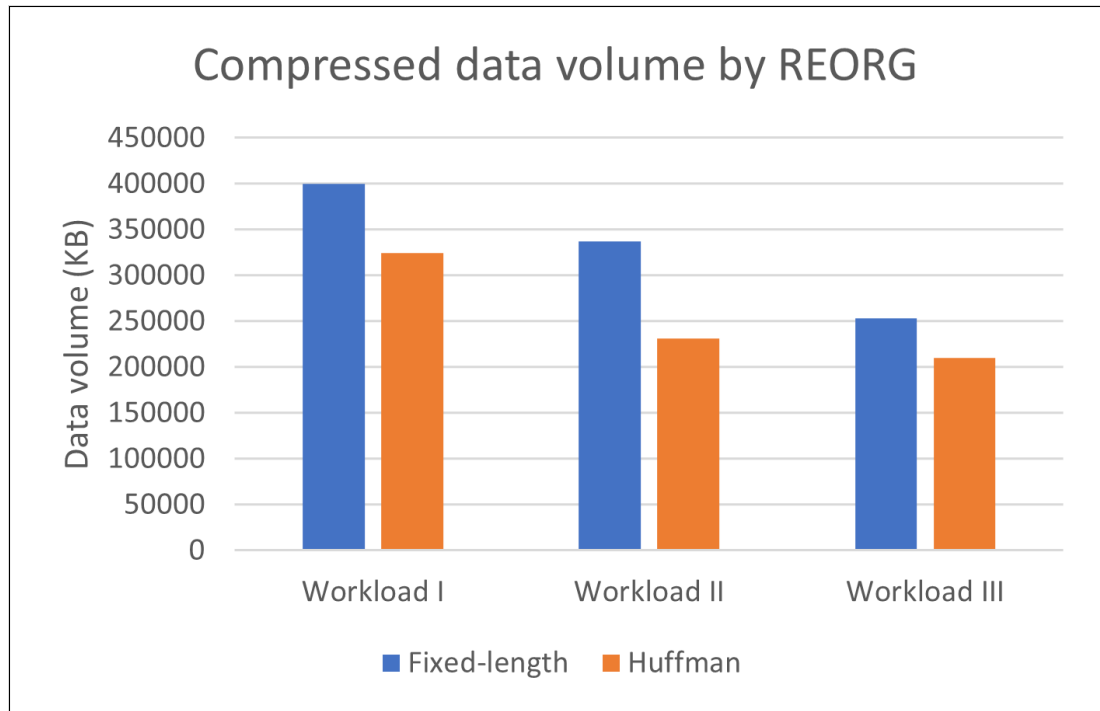


Figure 3-60 Compressed data size by using the REORG utility

### 3.11.3 Huffman compression performance

Performance evaluation of Huffman compression is described in the following two subsections. Unless specified otherwise, the same measurement environment was used to evaluate the performance of Huffman compression.

- ▶ Compression performance when using Db2 utilities
- ▶ Expansion (decompression) performance

#### Compression performance when using Db2 utilities

Using the Huffman algorithm to compress data does not add extra cost when compared to fixed-length compression. Figure 3-61 on page 117 shows the cost of using Huffman compression with **LOAD** and **REORG** during our measurements. Compared to fixed-length compression, Huffman compression shows a 1% - 5% improvement in both CPU and elapsed time.

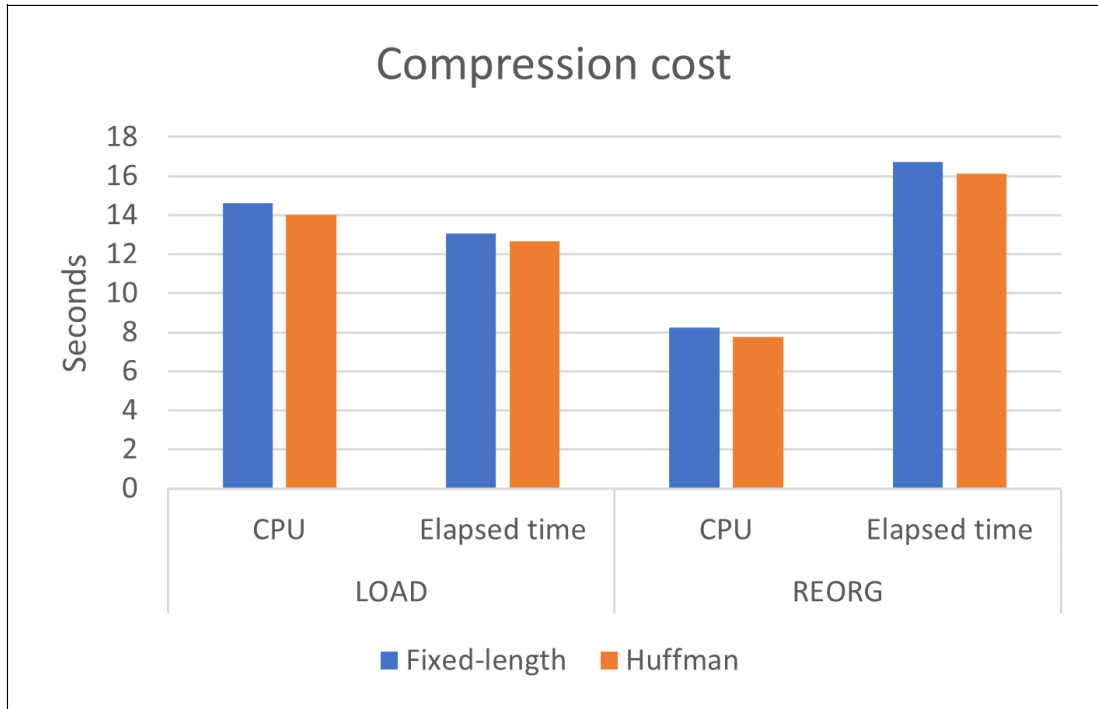


Figure 3-61 Compression performance by using LOAD and REORG

### Expansion (decompression) performance

When compressed data is used by an application, an extra CPU cost is incurred because the data must be expanded by using the compression dictionary. Conversely, using compression also can reduce the amount of getpage activity because fewer pages are needed to store the same number of rows. As a result, elapsed time can be reduced compared to when uncompressed data is used, especially when much data is processed.

### Comparing Huffman and fixed-length compression on IBM z14

To assess the overall performance impact of Huffman compression, performance regression buckets (PRBs) were selected, and the measurements were conducted in Db2 12 FL 508. For more information about the measurement environment and PRBs, see 4.1, “Performance regression bucket” on page 150.

The results showed no significant difference between using fixed-length compression and Huffman compression in terms of the overall elapsed time and total CPU time of the entire workload. We also observed that Huffman compression saved more data pages than fixed-length compression.

However, individual packages and jobs showed differences. The elapsed time difference ranges from -73% to 380%, and the CPU time difference ranges from -15% - 101%. Our tests revealed the following extra findings at the sub-workload level.

### Batch workload

The tables in the batch workload were populated through insert. The percentage of data pages that were saved for fixed-length compression is 59%, and for Huffman compression is 75%. The data in Table 3-17 shows an improvement in overall elapsed time of approximately 3%, and a slight CPU time degradation. For individual batch jobs, we observed these results:

- ▶ For sequential insert, there was an approximately 9% elapsed time improvement and a 15% CPU time improvement, which is due to a 15% reduction in page latch contention, a 25% reduction in global contention, and a 40% reduction in data buffer pool getpages.
- ▶ For random select, there was a 25% elapsed time improvement and a 9% CPU time improvement. We observed a 28% reduction in other read I/O and a 27% reduction in data buffer pool synchronous read I/Os.
- ▶ For piecewise delete, there was a 22% elapsed time improvement and a 9% CPU time improvement. We observed an approximately 50% reduction in service task switch (update commit) time and a 45% reduction in data buffer pool getpages.
- ▶ For pre-Db2 12 merge, there was a 73% elapsed time improvement and a 101% CPU time degradation. The large CPU time degradation for Huffman compression is because a table scan is selected, and fixed-length compression can leverage software partial decompression.
- ▶ For other cases, the difference was within 10% in elapsed time and CPU time.

Table 3-17 Batch workload

Compression	Class 1 elapsed time	Class 1 CPU time	Class 2 elapsed time	Class 2 CPU time
Fixed-length	5047.997536	1176.638761	4863.023989	1020.176169
Huffman	4887.32517	1189.147104	4703.383852	1033.039034
Delta (%)	3.18	1.06	3.28	1.26

### Utility workload

The tables were populated by using the **LOAD** utility. The number of data pages that were saved for fixed-length compression is 54% and for Huffman compression is 59%. The data in Table 3-18 shows a slight improvement in total elapsed time and no noticeable total CPU time difference for the overall utility workload.

Table 3-18 Overall utility workload

Utility workload	Fixed-length	Huffman	Delta (%)
Elapsed time	1352.82	1318.585	-2.53
CPU time	3929.18	3948.11	0.48

For individual utility jobs, we recorded the following observations, which are shown in Table 3-19 on page 119:

- ▶ For the **COPY** utility, we observed an approximately 10% elapsed time improvement and a 9% CPU time improvement. Improvement exists in both COPYR and COPYW phases, and data getpage activity was reduced by approximately 10%.
- ▶ For the **RECOVER** utility, we observed an approximately 12% elapsed time improvement and a 10% CPU time improvement. Improvement exists in the RESTORER, RESTOREW, and LOGAPPLY phases.
- ▶ For other utility jobs such as **LOAD**, **UNLOAD**, **REBUILD INDEX**, and **REORG**, no significant difference was observed in both elapsed time and CPU time.

Table 3-19 Utility type

Utility	Fixed-length	Huffman		Fixed-length	Huffman	
	Elapsed time	Elapsed time	Delta (%)	CPU time	CPU time	Delta (%)
LOAD	386.64	377.62	-2.33	2108.84	2122.51	0.65
UNLOAD	219.19	220.7	0.69	137.42	139.36	1.41
REBUILD INDEX	150.1	153.27	2.11	746.97	745.72	-0.17
RUNSTATS	70.24	71.16	1.31	67.2	68.09	1.32
COPY	44.92	40.23	-10.44	4.83	4.38	-9.32
RECOVER	167.9	147.33	-12.25	22.34	20.09	-10.07
REORG	313.83	308.275	-1.77	841.58	847.96	0.76

**OLTP workload**

We observed a slight elapsed time degradation and no noticeable CPU time difference, as shown in Table 3-20.

Table 3-20 OLTP workload

Compression	Class 1 elapsed time	Class 1 CPU time	Class 2 elapsed time	Class 2 CPU time
Fixed Length	0.082667	0.028139	0.078405	0.027510
Huffman	0.084556	0.028293	0.080754	0.027635
Delta (%)	2.28	0.55	3.00	0.45

**Query workload**

As shown in Table 3-21, we observed a slight total elapsed time degradation and no noticeable total CPU time difference. However, compared to fixed-length compression, the impact of Huffman compression on the elapsed time and CPU time performance of individual queries is significantly different and reveals a wide range of variation:

- ▶ Elapsed time changed in the range of -59% - 380%.
- ▶ CPU time changed in the range of -12% - 30%.

Table 3-21 Query workload

Compression	Class 1 elapsed time	Class 1 CPU time	Class 2 elapsed time	Class 2 CPU time
Fixed-length	2029.333025	2917.175441	2014.903341	2903.222642
Huffman	2092.768118	2932.002186	2078.33765	2918.053972
Delta (%)	3.13	0.51	3.15	0.51

**Table space scan example**

Among the query workloads, the worst-performing results that we observed from Huffman expansion were queries that use simple table scans. Although these types of queries are not representative of real workloads, we provide details about this scenario to demonstrate the difference in expansion behavior between Huffman and fixed-length compression.

The following queries all use a table scan so that we can evaluate the expansion cost of different types of queries. Data was compressed by using **REORG**, which resulted in an 83.5% savings for Huffman compression and an 81.7% savings for fixed-length compression.

Table 3-22 and Table 3-23 list the query types that were used during this test.

Table 3-22 Query type category

Query type	Description
Type A	Rscan 1314304366 rows, 1 column predicate, 0 row qualified
Type B	Rscan 1314304366 rows, 1 column predicate, 1000 rows qualified
Type C	Rscan 1314304366 rows, >2 column predicate, 0 row qualified
Type D	Rscan 1314304366 rows, no predicate, count(*)

Table 3-23 Query cases description

Type	Query case	Description
Type A	\$SEL15	Select column 15, where column 15=xx, 0 row qualified.
	\$SEL35	Select column 35, where column 35=xx, 0 row qualified.
	\$SELA	Select columns 1, 2, 3, 4, and 5, where column 19=xx, 0 row qualified.
Type B	\$SELB	Select columns 1, 2, 3, 4, and 5, where column 34=xx, 1000 rows qualified.
	\$SELE	Select columns 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10, where column 34=xx, 1000 rows qualified.
	\$SELG	Select columns 35, 36, 37, 38, and 39, where column 34=xx, 1000 rows qualified.
	\$SELH	Select columns 1, 2, 3, ... 30, 32, 33, and 34, where column 30=xx, 1000 rows qualified.
	\$SELJ	Select *, where column 30=xx, 1000 rows qualified.
Type C	\$SELC	Select columns 1, 2, 3, 4, and 5, where col9=aa, col15=bb, col19=cc, col35=dd, col39=ee, 0 row qualified.
	\$SELP	Select columns 1, 2, 3, 4, and 5, where col9=aa, col15=bb, col19=cc, col30=dd, col33=ee, col34=ff, col35=gg, col39=hh, 0 row qualified.
	\$SELQ	Select columns 1, 2, 3, 4, and 5, where col9=aa, col15=bb, col19=cc, 0 row qualified.
Type D	\$SELK	Select count(*), no predicate.

The results show that expanding Huffman-compressed data adds between approximately 4% and 65% extra CPU compared to fixed-length compressed data, and it saves approximately -3% - 38% elapsed time depending on the query type.

Figure 3-62 on page 121 and Figure 3-63 on page 121 compare the CPU cost and elapsed time that is associated with expansion for different types of queries for fixed-length expansion, Huffman expansion, and uncompressed data.



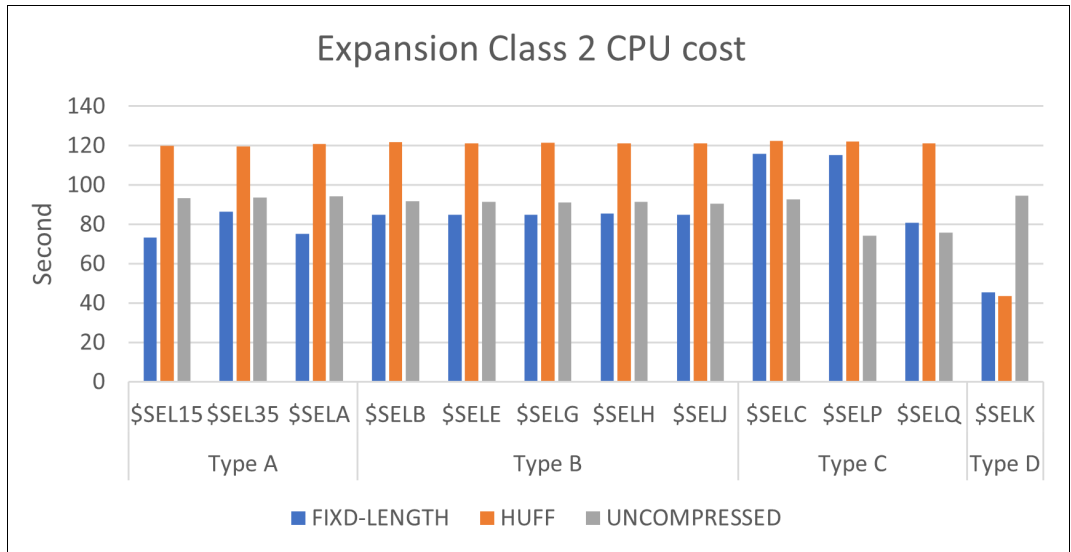


Figure 3-62 Db2 class 2 CPU time of table space scan queries

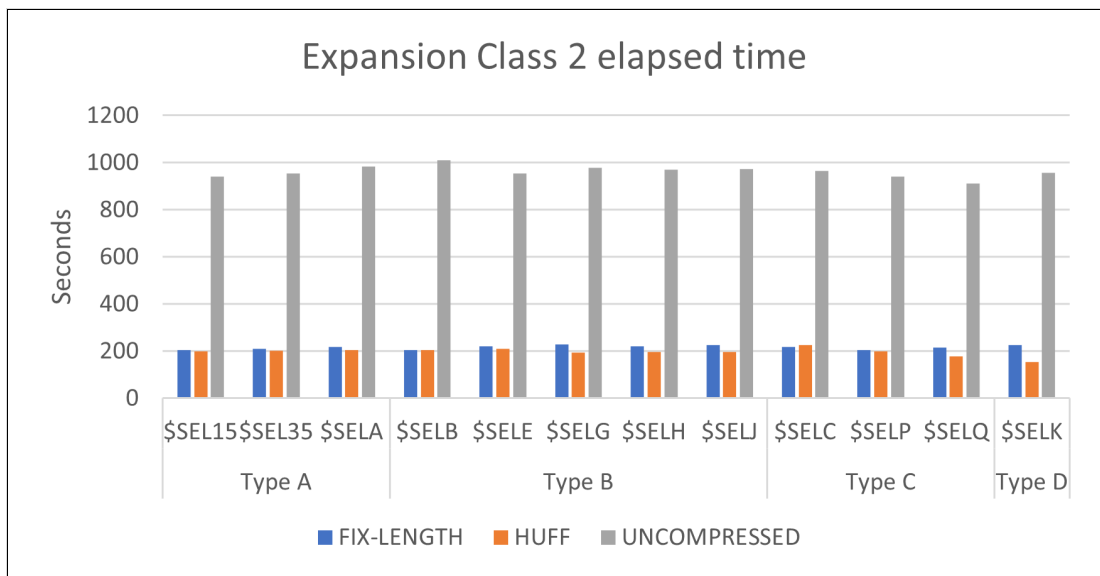


Figure 3-63 Db2 class 2 elapsed time of table space scan queries

To expand the objects that were compressed by the fixed-length algorithm, Db2 might use a technique called *partial expansion and decompression*. The partial decompression is effective against records with many columns because it expands only the necessary part of the record instead of expanding the entire record.

Db2 does not support partial expansion when the Huffman algorithm is used. Therefore, the CPU cost for different types of queries varies for fixed-length expansion (depending on the number of columns that are selected and the predicates evaluated), but it is consistent for Huffman expansion.

Figure 3-62 on page 121 and Figure 3-63 on page 121 show the expansion cost of the four types of queries with the different compression methods. All these queries use pure table space scan.

- Type A** In this type of query, a predicate evaluates one column and qualifies zero rows. This type of query is the worst case for full expansion. Db2 must decompress all rows, and none of the decompressed rows are needed. For this type of query, Huffman compression costs an average of 54% more CPU than fixed-length decompression. Elapsed time is reduced by 4% compared to fixed-length expansion for type A queries.
- Type B** In this type of query, a predicate evaluates one column and qualifies several rows. The CPU difference between Huffman and fixed-length compression is slightly smaller, with an average increase of 43%. Elapsed time is reduced by 8% on average compared with fixed-length expansion.
- Type C** In this type of query, a predicate evaluates more than two columns. The efficiency of partial expansion that is used by fixed-length expansion starts to drop when the column number increases in the predicates. The CPU increase for Huffman expansion is 6% for five column-predicated queries. Elapsed time is reduced by 15% on average compared with fixed-length expansion.
- Type D** This type of query is a count(\*) without predicate. Huffman-compressed data benefits from the fewer getpage requests due to the higher compression ratio. Therefore, compared with fixed-length compression, CPU and elapsed time are improved by 5% and 32%.

### Expansion performance by using IBM z13 and IBM z14

Although the Huffman algorithm can be used to compress data only on IBM z14 and later, it can be expanded on both IBM z13 and IBM z14 (and later). When data is Huffman-compressed (on IBM z14) but must be expanded on an older machine, such as IBM z13, you can use the EPROC. Expansion on machines older than IBM z14 is costly in terms of CPU usage because the variable Huffman index must be mapped into fixed-length compression.

To evaluate expansion cost, we evaluated same types of queries that are listed in the previous section. Our measurements show that Huffman expansion on IBM z13 requires 600% - 900% more CPU time than using fixed-length compression. A query that uses **count (\*)** without predicates is an exception because the data does not need to be expanded for the evaluation. In terms of elapsed time, Huffman expansion on IBM z13 requires 200% - 300% more elapsed time when compared with fixed-length expansion, but it saves approximately 10% elapsed time compared to uncompressed data.

Figure 3-64 on page 123 and Figure 3-65 on page 123 show the class 2 CPU and elapsed time respectively for these measurements.

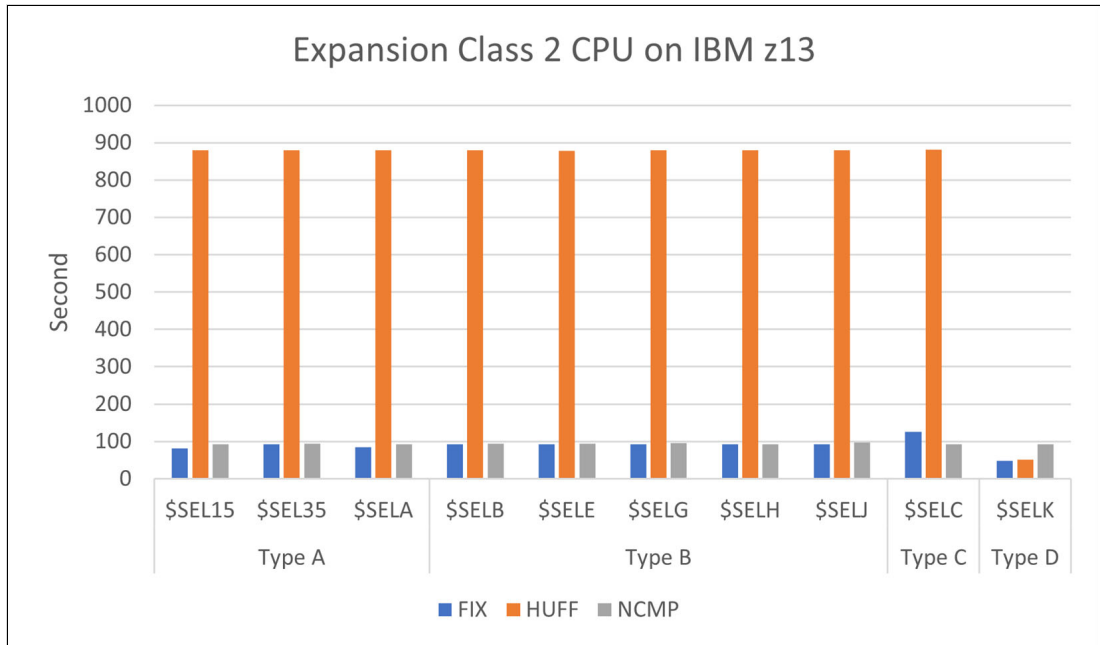


Figure 3-64 Db2 class 2 CPU time of table space scan on IBM z13

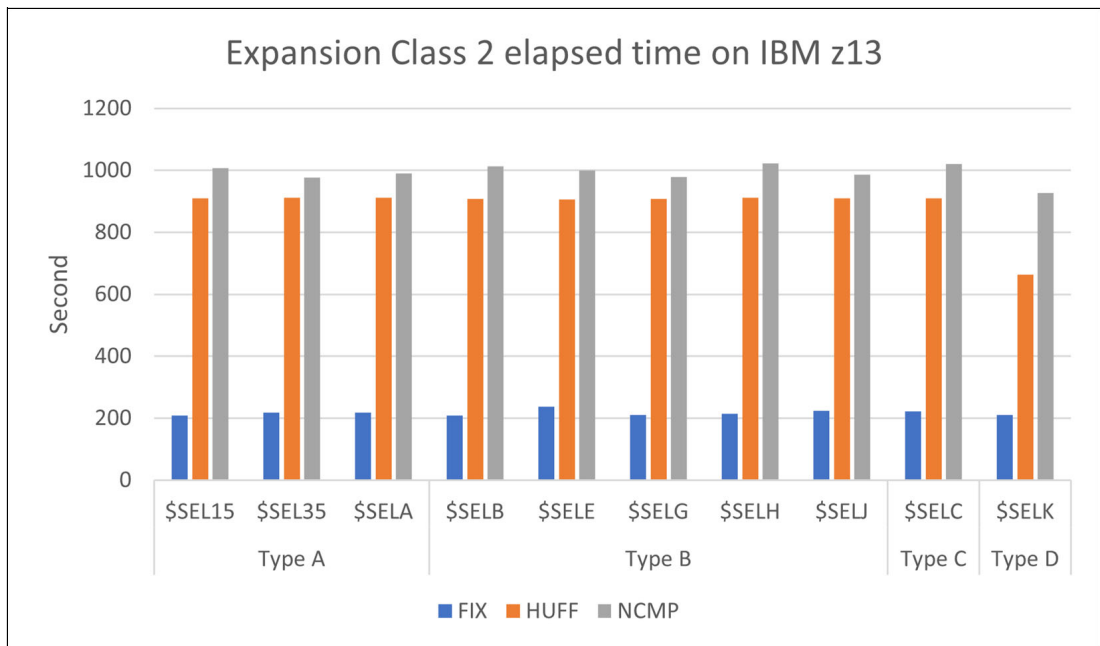


Figure 3-65 Db2 class 2 elapsed time of table space scan on IBM z13

As shown in Figure 3-65, four types of queries were tested. They all use a pure table space scan (rscan), which is the worst case scenario.

- ▶ Type A: A 924% increase in CPU and a 325% increase in elapsed time compared to fixed-length expansion on IBM z13.
- ▶ Type B: An 858% increase in CPU and a 316% increase in elapsed time compared to fixed-length expansion on IBM z13.

- ▶ Type C: A 597% increase in CPU and a 308% increase in elapsed time compared to fixed-length expansion on IBM z13.
- ▶ Type D: A 4% increase in CPU and a 215% increase in elapsed time compared to fixed-length expansion on IBM z13.

### **Performance measurements on IBM z15**

Similar performance measurements also were conducted with Db2 12 FL 504 by using IBM z15 hardware. The results are similar.

### **Performance measurements when using SAP core banking workloads**

The IBM Z performance team conducted a comprehensive performance evaluation of Huffman compression that used the SAP Account Settlement and Day Posting workloads that are part of SAP core banking workloads.

#### ***Disk space saving***

The measurements show that Huffman compression reduced the disk space that was used for the table spaces by 37% compared to fixed-length compression. The overall database space savings with Huffman compression are 18% because the workload includes many indexes that do not apply the savings from Huffman compression.

#### ***Performance comparison***

For the SAP Account Settlement workload (batch processing), the total batch elapsed time was shortened by 6%, and the database server ITR improved by 3% for Huffman compression versus the fixed-length baseline. For the SAP Day Posting workload (OLTP), the average response times are in the subsecond range and comparable for both encoding techniques, but the database server ITR was down by 4% for the Huffman compression versus the fixed-length baseline.

For more information about this measurement, see [Performance Evaluation of Huffman Compression with SAP Banking on IBM Z](#).

## **3.11.4 Performance considerations**

When considering whether to use Huffman compression, account for the following factors:

- ▶ Compression ratio
- ▶ Performance of expansion
- ▶ Avoiding expansion on IBM z13

### **Compression ratio**

We observed better compression ratio when using the **REORG** utility as opposed to the **INSERT** or **LOAD** utilities. Consider running the **REORG** utility to gain the full benefit of Huffman compression.

### **Performance of expansion**

Our measurements indicate that the performance of expansion highly depends on the data access pattern and the compression ratio. In a typical OLTP environment with heavy index access, the cost of the data page decompression tends to be small, and you do not see a significant difference between Huffman and fixed-length compression.

However, in cases where SQL statements can benefit from the partial decompression feature that was introduced in Db2 11 (such as selecting a few columns and scanning many data pages with simple predicates), we observed that fixed-length compression is more efficient than Huffman because Huffman does not support partial decompression; it always expands the entire row to do predicate evaluation. Therefore, the CPU cost to expand Huffman-compressed data can be higher than for fixed-length compression for some types of queries (the types that can leverage partial decompression).

For most of the queries that we tested, the elapsed time was reduced because fewer getpage and prefetch requests were needed when processing large amounts of data, such as when performing a large table space scan. To estimate the impact of transforming fixed-length compression into Huffman compression, you can inspect the query types and their access paths in your applications to see which query type or types to which they belong.

### Avoiding expansion on IBM z13

The test results of expansion on IBM z13 show that it costs more CPU to expand Huffman-compressed records on IBM z13 because IBM z13 does not support Huffman hardware decompression. Using the EPROC to do the software transformation is CPU-intensive, so avoid querying Huffman-compressed data on IBM z13.

## 3.11.5 Using DSN1COMP for Huffman compression estimation

You can use the **DSN1COMP** utility to estimate how efficient compression will be when you plan to use **COMPRESS YES** for a table space. Db2 12 FL 504 enabled the usage of the Huffman compression algorithm to achieve a better compression ratio. **DSN1COMP** also was enhanced to provide an estimation for Huffman compression by specifying the **TYPE HUFFMAN** or **ALL** option. Figure 3-66 shows an example of specifying **ALL**. **DSN1COMP** produces a compression estimation in a table for both fixed-length and Huffman compression.

```

DSN1940I DSN1COMP COMPRESSION REPORT
HARDWARE SUPPORT FOR HUFFMAN COMPRESSION IS AVAILABLE
+-----+-----+-----+-----+
|                                     | UNCOMPRESSED | COMPRESSED | COMPRESSED |
|                                     |              | FIXED      | HUFFMAN     |
+-----+-----+-----+-----+
| DATA (IN KB)                       |          1,709,306 |          651,692 |          599,970 |
| PERCENT SAVINGS                      |                   |             61% |             64% |
| AVERAGE BYTES PER ROW               |             170 |             67 |             61 |
| PERCENT SAVINGS                      |                   |             60% |             64% |
| DATA PAGES NEEDED                   |          473,575 |          182,799 |          165,396 |
| PERCENT DATA PAGES SAVED           |                   |             61% |             65% |
| DICTIONARY PAGES REQUIRED              |             0 |             16 |             20 |
| ROWS SCANNED TO BUILD DICTIONARY     |                   |             824 |             824 |
| ROWS SCANNED TO PROVIDE ESTIMATE     |                   | 10,418,629 | 10,418,629 |
| DICTIONARY ENTRIES                   |                   |           4,096 |           4,080 |
| TOTAL PAGES (DICTIONARY + DATA)     |          473,575 |          182,815 |          165,416 |
| PERCENT SAVINGS                      |                   |             61% |             65% |
+-----+-----+-----+-----+
DSN1994I DSN1COMP COMPLETED SUCCESSFULLY,          473,855 PAGES PROCESSED

```

Figure 3-66 DSN1COMP output of compression statistic by using the ALL option

To use **DSN1COMP** to estimate the efficiency of Huffman compression in Db2 12, you must have the PTF for APAR PH19242 applied.

In addition, apply the PTF for APAR PH34808. This APAR enables you to run **DSN1COMP** against an object that is compressed with fixed-length compression to estimate how much extra compression that you might achieve by using Huffman compression instead.

### Huffman compression ratio estimation

Using **LOAD** or **REORG** often results in different compression ratios. **REORG** can often compress data more efficiently because Db2 samples more data before creating the dictionary.

By default, **DSN1COMP** provides estimates like the compression savings that the **LOAD** utility would achieve. If you specify the **REORG** keyword, **DSN1COMP** provides estimates like the compression savings that the **REORG** utility would achieve.

To avoid spending too much time and processor resources on compression estimation, **DSN1COMP** provides a **ROWLIMIT** option to limit the number of rows that are scanned. A sample of the first 5 - 10 million rows of the table space or partition suffices. You can calculate the proper **ROWLIMIT** based on the row length of the table for which you are doing the estimation.

### Evaluating DSN1COMP compression estimations

We tested **DSN1COMP** on 13 tables that contain large data volumes, with options to simulate **LOAD** and **REORG**, and with and without **ROWLIMIT**. Table 3-24 and Table 3-25 on page 127 show the Huffman compression savings (green column) and the **DSN1COMP**-estimated Huffman compression savings (blue column) after scanning all rows. The yellow column is the **DSN1COMP**-estimated Huffman compression savings after scanning 10 million rows of data.

Table 3-24 *DSN1COMP estimated and actual savings with the LOAD option*

Table	Original total data size (KB)	Actual data size after compression (KB)	Actual savings	DSN1COMP estimated savings (scanned all rows)	Actual data size scanned with limited rows (KB)	DSN1COMP estimated savings (scanned limited rows (10 million rows scanned))
TB1	1709306	585665	66%	65%	9,651	65%
TB2	1723378	345087	80%	79%	9,756	82%
TB3	703792	232143	67%	66%	9,754	67%
TB4	2054482	224012	89%	89%	9,755	89%
TB5	1089010	600927	45%	47%	9,420	51%
TB6	1508554	267890	82%	76%	9,772	81%
TB7	852313	485027	43%	45%	9,728	63%
TB8	2961256	151667	95%	95%	9,765	85%
TB9	1005218	257356	74%	71%	9,936	76%
TB10	642851	300910	53%	52%	11,163	50%
TB11	497195	202357	59%	57%	11,016	52%
TB12	259038	259030	0%	0%	9,050	0%
TB13	498974	405511	19%	16%	10,515	79%

Table 3-25 DSN1COMP estimated and actual savings with the REORG option

Table	Original total data size (KB)	Actual data size after compression (KB)	Actual savings	DSN1COMP estimated savings (scanned all rows)	Actual data size scanned with limited rows (KB)	DSN1COMP estimated savings (scanned limited rows)
TB1	1709306	603309	65%	65%	26,042	66%
TB2	1723378	314356	82%	79%	38599	83%
TB3	703792	200039	72%	66%	26168	66%
TB4	2054482	198663	90%	89%	9,755	91%
TB5	1089010	405259	63%	47%	9,420	61%
TB6	1508554	262668	83%	76%	9,772	83%
TB7	852313	240513	72%	45%	9,728	69%
TB8	2961256	131080	96%	95%	9,765	97%
TB9	1005218	236378	76%	71%	9,936	77%
TB10	642851	252452	61%	52%	11,163	53%
TB11	497195	140637	72%	57%	11,016	53%
TB12	259038	258996	0%	0%	9,050	0%
TB13	498974	188420	62%	16%	10,515	81%

These results show that **DSN1COMP** provides an accurate estimate of the data compression savings even when the **ROWLIMIT** option is used.

TB13 is an exception, which might because of the data distribution and data characteristics. TB12 is not compressed because the data was encrypted by the application.

### 3.12 Encrypting Db2 data with z/OS data set encryption and CF structure encryption

Data security is a core part of the modern IT industry. Db2 for z/OS takes every opportunity to improve its data protection capabilities. When z/OS DFSMS introduced the pervasive encryption of data-at-rest feature that is called *z/OS data set encryption*, Db2 for z/OS was among the first products to leverage this new feature and offered users the option to encrypt transparently Db2 data sets. When a data set is encrypted, unauthorized users cannot view the contents of the data set in clear when it is on the disk.

You can encrypt the following types of Db2 data sets:

- ▶ Table space and index space data sets.
- ▶ Active and archive log data sets.
- ▶ Catalog and directory data sets.
- ▶ Data sets that are used by Db2 online utilities, for example, temporary work files, data files for loading and unloading, image copy data sets, and so on.
- ▶ Input and output data sets for stand-alone utilities such as **DSN1LOGP**.

On systems that run z/OS 2.3 or later, CF list and cache structure entries and entry-adjunct data can be encrypted while the data is being transferred to and from the CF and the data is in the CF structure. For Db2, this capability means that data in the GBP and SCA CF structures can be encrypted on a structure-by-structure basis.

### 3.12.1 Requirements

Before you can start using DFSMS data set encryption with Db2 related data, your system must meet the hardware and software requirements.

#### Data set encryption

To use data set encryption for Db2-related data sets, your system must meet the following requirements:

- ▶ Hardware requirements:
  - IBM Enterprise z196 or later
  - Crypto Express3 Coprocessor or later
  - Feature 3863 CP Assist for Cryptographic Functions (CPACF)
- ▶ z/OS 2.3 and z/OS 2.4 (with OA56622) or later (z/OS 2.2 with APARs OA50569 and OA53951 also is supported).
- ▶ ICSF installed and configured with a CKDS and AES master key loaded.
- ▶ Db2 12 or later. Using key label management for z/OS DFSMS data set encryption in Db2 requires Db2 12 FL 502 and later.

For more information about requirements and considerations for encrypting Db2 data sets, see [Encrypting your data with z/OS DFSMS data set encryption](#).

#### CF structure encryption

To encrypt Db2 GBP and SCA data in the CF, your system must meet the following requirements:

- ▶ Hardware requirements:
  - IBM zEnterprise EC12 or IBM zEnterprise BC12 or later servers
  - Crypto Express3 Coprocessor or later
  - Feature 3863 CPACF
- ▶ z/OS 2.3 and later.
- ▶ ICSF installed and configured with a CKDS and AES master key loaded.
- ▶ Db2 version: All supported versions.

### 3.12.2 Performance measurements for data set encryption

As you might expect, using data set encryption incurs extra costs. For Db2 workloads, the cost is reflected in different areas at run time, depending on the types of data sets that are being encrypted. This section provides some guidance about where to look to identify the cost of encryption after you start encrypting your Db2 data sets. It also provides some results from our measurements to give you an idea about the CPU overhead and elapsed time increases that we observed in our configurations. These measurements should give you a rough idea what kind of performance impacts you might see when you encrypt your Db2 data sets.



Here are a few rules to help you understand data encryption overhead:

- ▶ Data set encryption and decryption happen only when I/O operations are performed.
- ▶ Data set encryption and decryption are performed at the I/O buffer level:
  - For VSAM data sets, the encryption unit is a CI.
  - For sequential data sets, the encryption unit is a block.

To illustrate how to identify the *data decryption* cost, if you have a table space with a data set that is encrypted and your workload does synchronous database I/Os against this table space, then each database I/O requires data decryption when the data is read from disk into the Db2 local buffer pool. Because data decryption for VSAM data sets is part of the I/O interrupt handling, the CPU cost is accounted for under the *ssnmDBM1* address space. By looking at the statistics report formatted by OMPE, you are likely to see an increase in “CPU FOR I/O” time (QWSAIIPT) for the database services (*ssnmDBM1*) address space in the “CPU TIMES 2” trace block.

To explain to where the *data encryption* cost goes, take active logs, for example. If your active logs are created as encrypted data sets, then each log write I/O causes the corresponding log CIs to be encrypted on writing to the disk. Because the system services (*ssnmMSTR*) address space is responsible for log write activities, this data encryption cost is accounted for as a part of *ssnmMSTR* PREEMPT SRB time and is zIIP eligible.

The different elapsed time and CPU time components that are affected by the usage of data set encryption and the instrumentation that is available to assess the impact of reading/writing encrypted Db2 data sets and log data sets, are described in more detail next.

## Measurement environment

A series of tests was designed to evaluate the overhead of running workloads with various types of encrypted data sets, including table space data sets, log data sets, utility data sets, and so on. The data set encryption cost is related to I/O operations, so the actual cost that you see for your workloads depends on the I/O characteristics of the data sets that you choose to encrypt, and they also depend on your hardware configuration.

We used the following basic configurations to obtain our measurements:

- ▶ IBM z13 and IBM z14 processors
- ▶ Two Crypto6S crypto cards per LPAR
- ▶ 512 GB of LPAR storage (For two-way data-sharing tests. Two LPARs each has 512 GB of storage.)
- ▶ DS8870 disk controller
- ▶ CF LEVEL 22 for data-sharing tests
- ▶ z/OS 2.3

## Measurement scenario description

To provide a clear picture of the basic encryption cost, the performance evaluation tests were designed in such a way that each one of them targets different data set types and I/O behaviors. They all invoke heavy I/O operations to trigger a high level of data encryption and decryption activities so that we can measure the cost more effectively. The measurement results of the tests are in a sense worst-case scenarios in terms of data encryption cost because of their higher-than-normal I/O activities.

Table 3-26 lists the important characteristics of the workloads that we used for evaluating the encryption overhead for each type of data set.

Table 3-26 Data set encryption special test workload characteristics

Data set I/O operations	Workload used for evaluation	Encrypted data sets	Number of I/O operations (million)	Number of CPs General CP or zIIP
DATABASE I/O 4 K pages	Random Select	Table space	240.6	8/0
DATABASE I/O 32 K pages	Random Select	Table space	295.3	8/0
PREFETCH 4 K pages	Full table space scan	Table space	19.3	8/3
DEFERRED WRITE 4 K pages	Multiple-row insert (MRI)	Table space	6.1	4/3
DEFERRED WRITE 32 K pages	MRI	Table space	0.8	4/3
CASTOUT 4 K pages	MRI	Table space	12.8	4+4/3+3 (two-way data sharing)
ACTIVE LOG WRITE	MRI	Active log	18.5	4/3
ACTIVE LOG READ	Log offload	Active log	0.9	4/3
ARCHIVE LOG WRITE	Log offload	Archive log	0.9	4/3
ARCHIVE LOG READ 8 K BLKSIZE	Large UR rollback	Archive log	3.2	4/3
ARCHIVE LOG READ 24 K BLKSIZE	Large UR rollback	Archive log	1.1	4/3

## Measurement results

This section presents the measurement results that can help you understand the changes that you might see in different performance areas after implementing data set encryption.

### Data decryption: Database I/O

The random select workload measurements with heavy synchronous database I/O (DB I/O) were run by using IBM z14 hardware. When the table space data set is encrypted, we observed the following results:

- ▶ DB I/O operations have slightly longer average response times, resulting in longer class 2 elapsed time compared to when the table space data set is not encrypted:
  - For 4 K pages, the average database I/O suspension time increased by 0.46%.
  - For 32 K pages, the average database I/O suspension time increased by 2.01%.
- ▶ Class 2 CPU time was reduced compared to when the table space data set is not encrypted with similar volume of DB I/O requests:
  - For 4 K pages, the Class 2 CPU time is 0.91% less.
  - For 32 K pages, the Class 2 CPU time is 0.51% less.

- ▶ DBM1 I/O interrupt time is where most of the CPU cost is accounted for data decryption. The page size determines the DB I/O operation's data decryption CPU cost: the bigger the page size, the higher the CPU cost for data decryption per I/O.
  - For 4 K pages, *ssnm*DBM1 I/O interrupt time increased by ~0.001 ms per database I/O.
  - For 32 K pages, *ssnm*DBM1 I/O interrupt time increased by ~0.006 ms per database I/O.
- ▶ Considering the total Db2 CPU usage, which is made up of class 2 CPU and all Db2 system address space CPU time, the extra CPU cost of encryption with this workload is 7% and 43% for 4 K and 32 K pages.

Figure 3-67 shows the performance measurement numbers that reflect these observations.

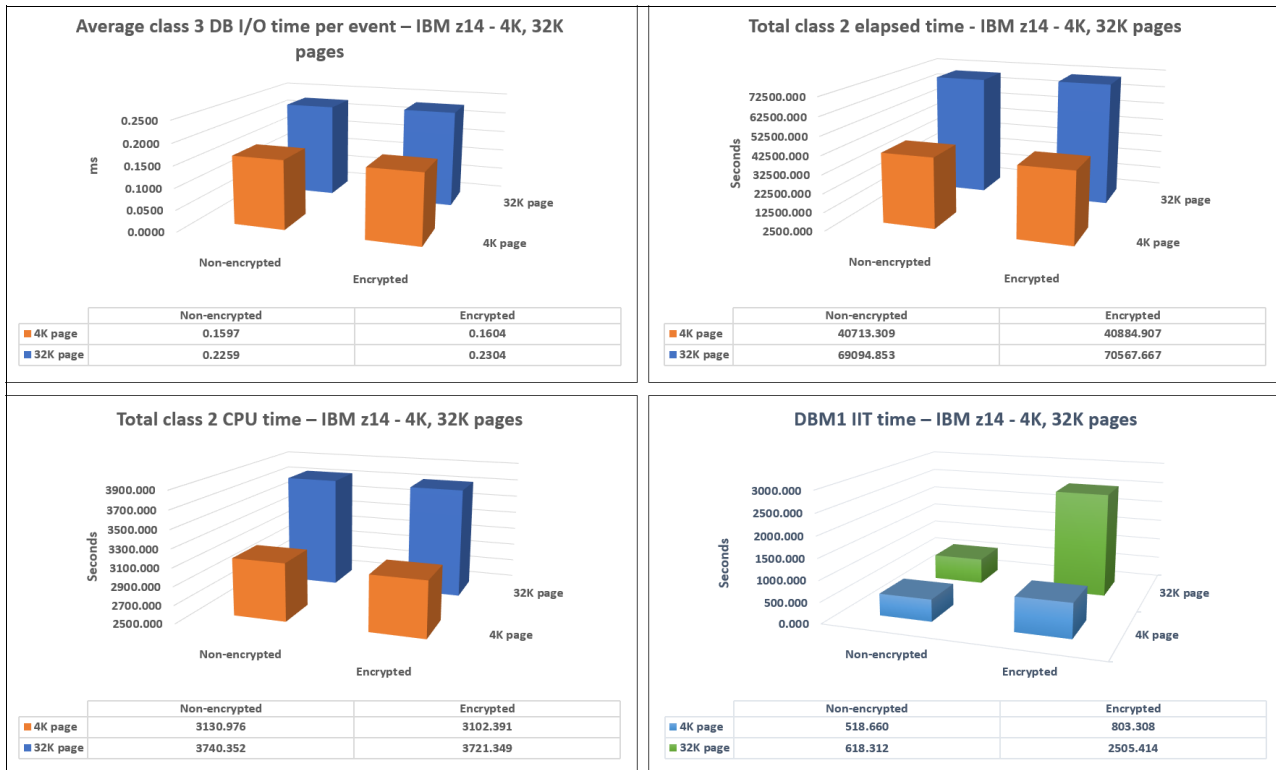


Figure 3-67 Data decryption (DB I/O) for 4 K and 32 K pages

As the first generation of IBM zSystems with pervasive encryption capability, IBM z14 has much better data encryption efficiency than IBM z13. By comparing the performance results of the same random select workload running on IBM z13 and IBM z14, as shown in Figure 3-68, we can see following performance results:

- ▶ For 4 K pages, IBM z14 uses 40% of the CPU time (DBM1 IIT time) that IBM z13 spends to decrypt a similar number of pages for synchronous database I/Os.
- ▶ For 32 K pages, IBM z14 uses only 31% of the CPU time (DBM1 IIT time) that IBM z13 spends to decrypt a similar number of pages for synchronous database I/Os.

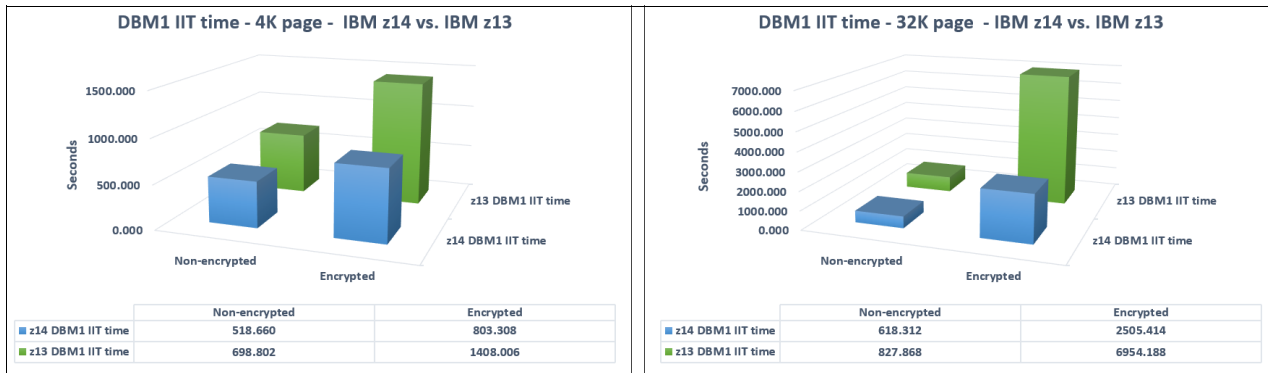


Figure 3-68 Data decryption: IBM z14 versus IBM z13

### Data decryption: Prefetch

We used a sequential prefetch workload to measure decryption performance for data prefetching. The base table of the workload has 1000 partitions and uses a 4 K page size. The SQL statement is a **SELECT** statement with a **WHERE** clause on a non-indexed column that has an equal predicate of a nonexistent value. This setup triggers a full table space scan. The query was run with a parallel degree of 16 on an IBM z14 processor.

By using these measurements and comparing the scenarios in which the table space data sets were encrypted versus not encrypted, we made the following observations:

- ▶ OTHER READ time increased by as much as 28%. This increase caused class 2 elapsed time to increase by approximately 11%.
- ▶ Most of the decryption CPU cost for sequential prefetch goes to the *ssnm*DBM1 I/O interrupt time, and the increase is as large as 13 times. Concurrently, the *ssnm*DBM1 preemptive SRB time shows a small reduction.
- ▶ The total Db2 CPU time, including class 2 CPU and all Db2 address spaces CPU time, increased by approximately by 24%.

Figure 3-69 on page 133 shows the performance measurement numbers that reflect these observations.

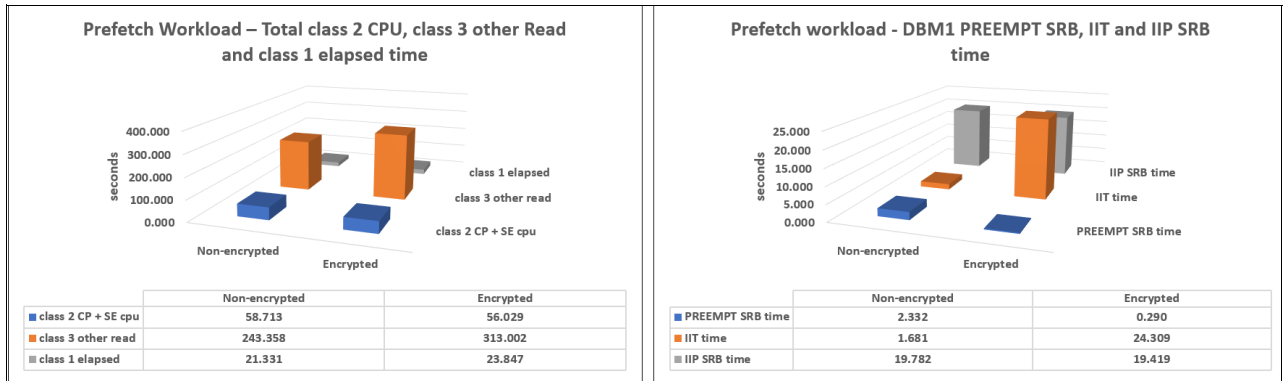


Figure 3-69 Data decryption (prefetch) of 4 K pages on IBM z14

**Data encryption: Deferred write and castout**

When a data set is encrypted with z/OS data set encryption, data encryption occurs when data is written out to disk. For Db2 table space and index data sets, the result is that data encryption happens either during deferred write or cast-out operations. We used an MRI workload to evaluate the encryption cost of user data in a non-data-sharing environment for deferred writes, and in a two-way data-sharing environment for cast-out operations. The tests were run on an IBM z14 processor.

From the measurement results, as shown in Figure 3-70 and Figure 3-71 on page 134, we observed the following facts that are related to the encryption cost with data set encryption for Db2 user data.

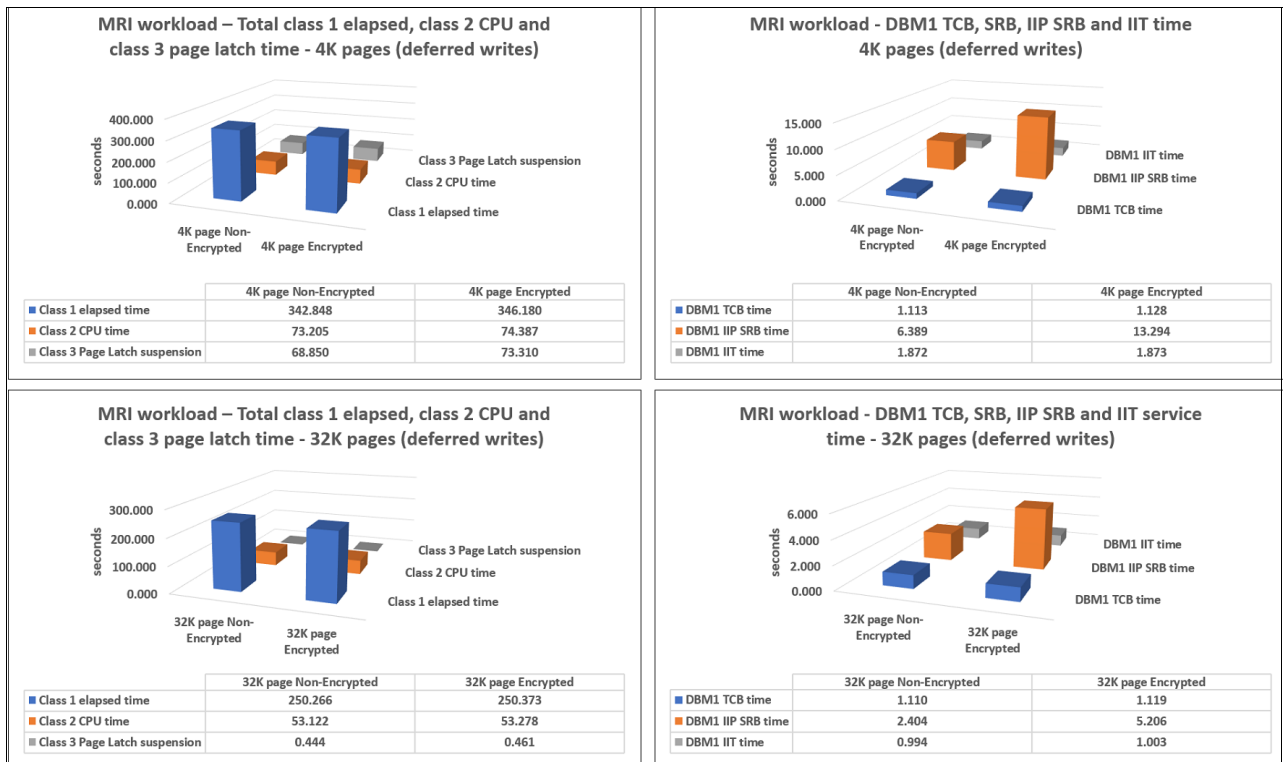


Figure 3-70 Data encryption (deferred writes): MRI with 4 K, 32 K pages

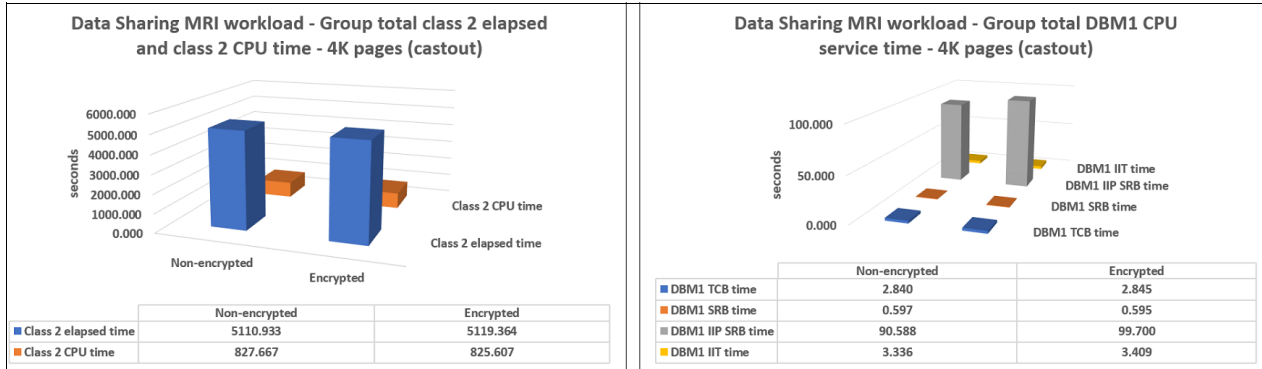


Figure 3-71 Data encryption (castouts): MRI with 4 K pages

- ▶ The data encryption CPU cost is registered under *ssnmDBM1* SRB time and is zIIP eligible for both deferred write and cast-out operations.
- ▶ As we were looking at deferred write performance, our test triggered heavy deferred write activity. This heavy deferred write activity resulted in long page latch suspension for the workload. When the data is encrypted, deferred writes must do extra work to encrypt the data, so longer class 3 suspension time for existing “pressure points” (such as page latch suspension time in our scenario) should be expected when the underlying data sets are encrypted.
- ▶ Encrypting 32 K pages costs more CPU per page than encrypting 4 K pages because 32 K CIs are being encrypted for each write:
  - As shown in Table 3-26 on page 130, the 4 K deferred write test writes 6.1 million pages, and the *ssnmDBM1* IIP SRB time increase is 6.905 seconds for data encryption, which calculates to approximately 1.13 seconds per 1 million pages written.
  - The 32 K deferred write test writes 0.8 million pages, and the *ssnmDBM1* IIP SRB time increase is 2.802 seconds for data encryption, which calculates to approximately 3.5 seconds per 1 million pages written.

### Log encryption and decryption: Active logs

To evaluate the data encryption cost of active logs, we used an MRI workload to produce mass active log write operations. For measuring the active log data decryption cost, we designed a test that offloads 3.5 GB of log records from encrypted active logs to non-encrypted archive logs. By comparing the CPU cost and workload elapsed time differences between tests that are done with non-encrypted active log data sets and encrypted active log data sets, we analyzed the cost of active log data encryption and decryption.

We made the following observations based on the test results:

- ▶ For active log write operations:
  - The data encryption cost for active log writes is charged to *ssnmMSTR* SRB time and is zIIP eligible.
  - When the active log data sets are encrypted, the update commit suspension time increases because data encryption causes log writes to take longer to complete, which causes longer class 2 elapsed time for the MRI workload that is using encryption for the active log data sets.

Figure 3-72 on page 135 shows the detailed performance numbers of the active log encryption tests from our MRI workload.

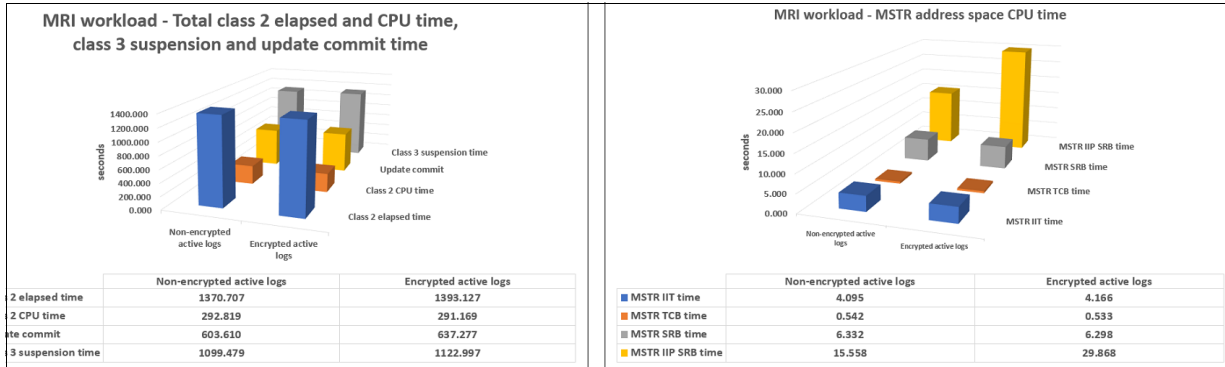


Figure 3-72 Active log encryption with MRI

- For active log read operations with offload

The CPU cost of active log data decryption during offloading is accounted for under the I/O interrupt time of the *ssnm* MSTR address space. Although percentage-wise the CPU increase is many-fold, the absolute increase to offload 0.9 million log CIs is small enough to ignore (a mere 0.6 second), as shown in Figure 3-73.

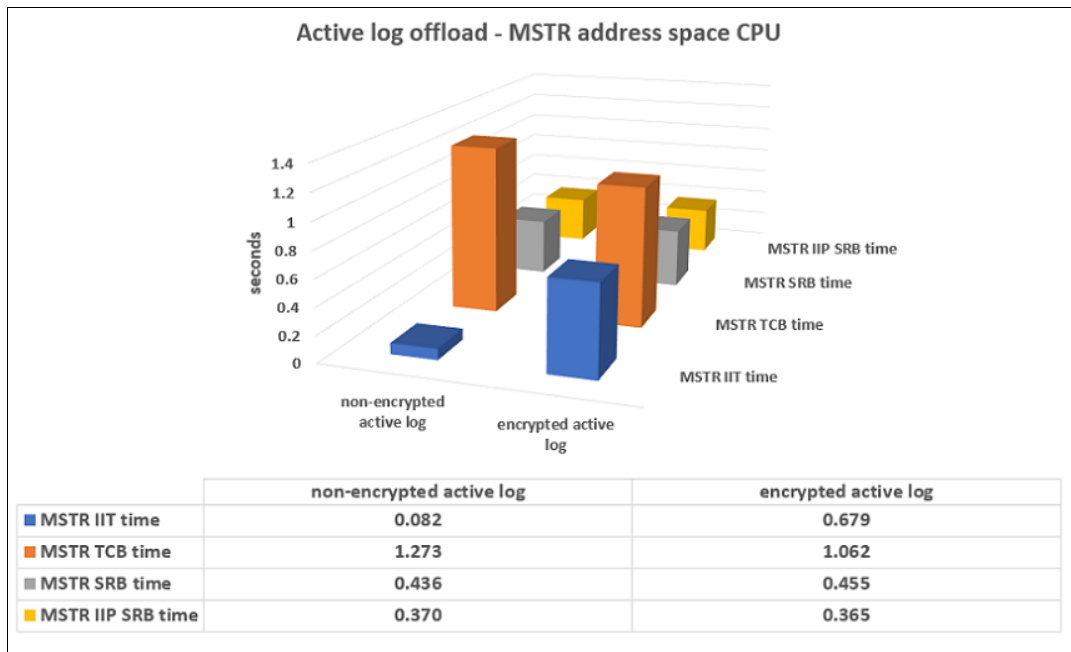


Figure 3-73 Active log decryption, log offload: MSTR address space CPU

### Log encryption and decryption: Archive logs

To evaluate archive log data set encryption behavior, we used the same test that offloads 0.9 million CIs of log records from non-encrypted active logs to archive logs.

For archive log data set decryption, we designed a large rollback workload that reads 3.5 GB of archive logs. We first ran an MRI batch workload without committing to fill a 4 GB active log to 88% full, and then we suspended the program. Next, we issued the **-ARCHIVE LOG** command to offload the log data to archive logs and made sure that the data was no longer available on any active log data set. After that, we canceled the suspended MRI batch job, which required the rollback to read log data from the archive logs.

By comparing the performance data between non-encrypted archive log data sets and encrypted archive log data sets during the tests, we learned the following facts about encrypting and decrypting Db2 archive log data sets:

- ▶ Both data encryption and decryption are performed by the *ssnm*MSTR address space that is charged as NONPREMPT SRB time when accessing encrypted archive logs. This CPU time is not zIIP eligible. The absolute CPU usage increase is small enough to ignore.
- ▶ The data decryption CPU cost for archive log data sets when accessed for rollback is less with 24 K BLKSIZE than 8 K BLKSIZE to retrieve the same number of log records.
- ▶ The elapsed time of the active log offload and rollback from log records in the archive log are not affected when running against encrypted archive logs.

The measurement details are shown in Figure 3-74, Figure 3-75, and Figure 3-76 on page 137.

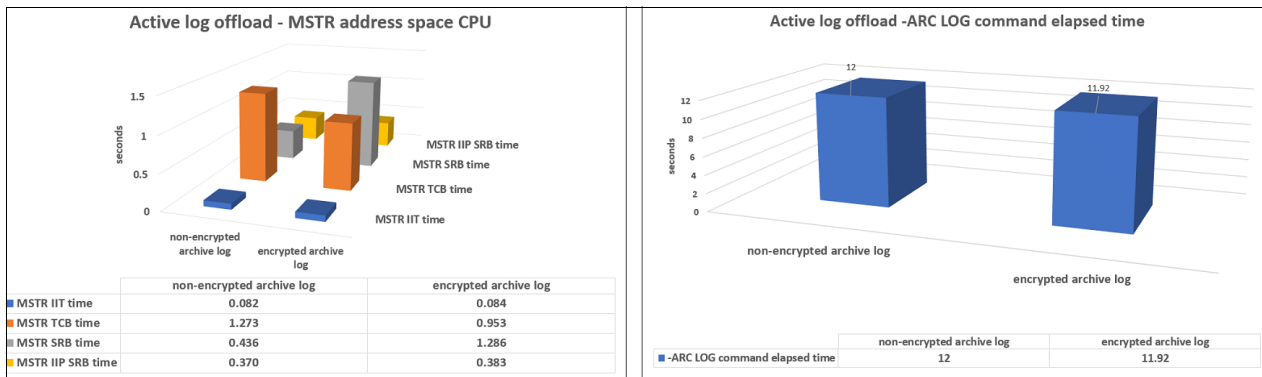


Figure 3-74 Archive log encryption, log offload: CPU and elapsed

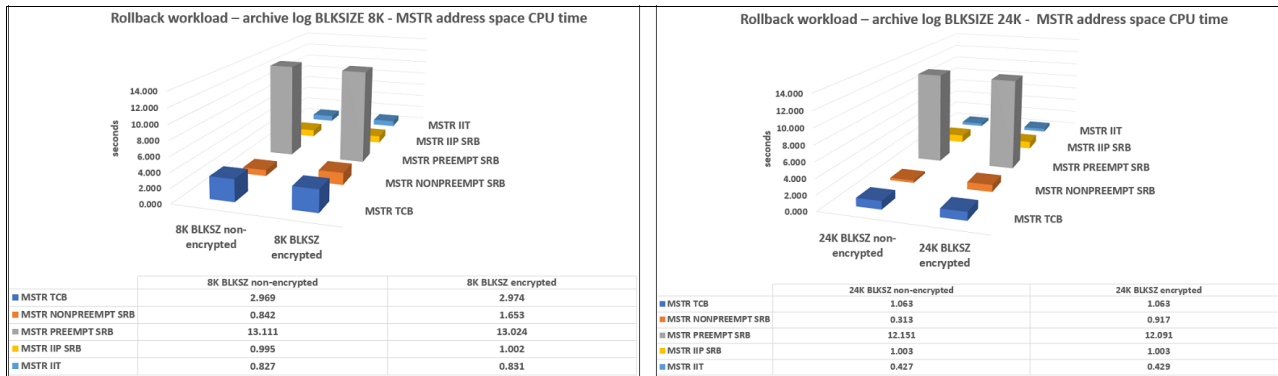


Figure 3-75 Archive log decryption, rollback: BLKSIZE 8 K and 24 K CPU



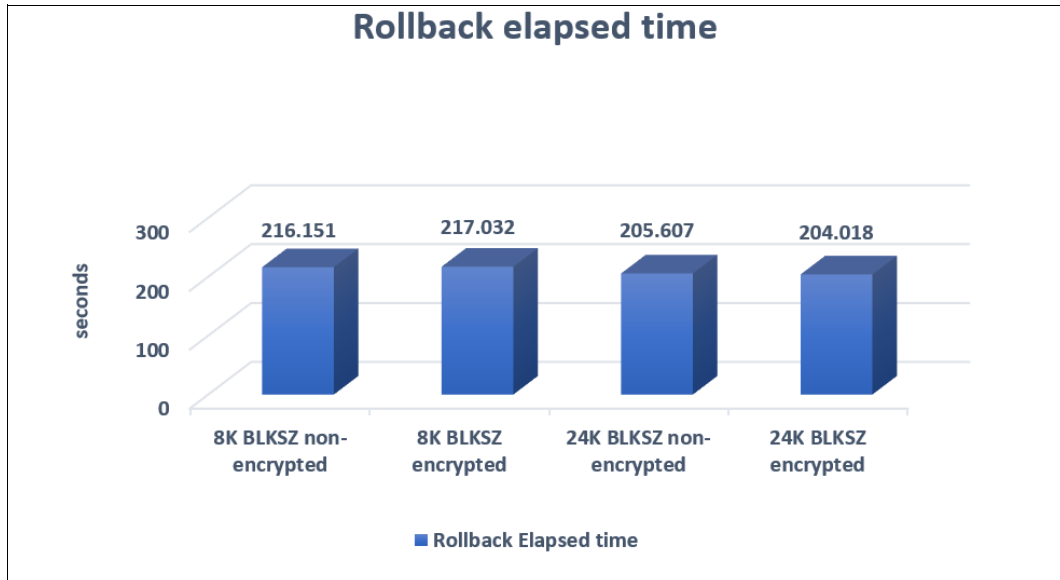


Figure 3-76 Archive log decryption, rollback: BLKSZ 8 K and 24 K elapsed

**Note:** The rollback elapsed time is calculated as the elapsed time between the cancel job command and job cancel success time.

### Summary of results

Let us summarize the knowledge that we gained so far about the encryption and decryption cost of running Db2 workloads with different types of Db2 data sets encrypted.

- ▶ Encrypting user and catalog table spaces and indexes:
  - Data decryption for table spaces and index spaces happens during the read I/Os, such as synchronous database I/Os and prefetch I/Os, and the CPU cost is accounted for as part of the *ssnmDBM1* address space's I/O interrupt time.
  - Data encryption for table spaces and index spaces occurs during the write I/Os, such as deferred writes and GBP cast-out operations, and the data encryption cost is accounted for as part of *ssnmDBM1* PRERMPPT SRB time and is zIIP eligible.
- ▶ Encrypting Db2 logs
  - For active logs, writing to encrypted log data sets causes the *ssnmMSTR* address space to spend more zIIP eligible PREEMPT SRB time to encrypt the log CIs. When Db2 must read active logs for offloading, rollbacks, recovery, and so on, the *ssnmMSTR* address space decrypts the log CIs, and the CPU cost is accounted for as part of the I/O interrupt service time of the *ssnmMSTR* address space.
  - For archive logs, encryption happens during the offload process, and decryption happens when operations such as rollback and recovery must read log records from (encrypted) archive logs. The archive log encryption cost is charged to MSTR NONPREEMPT SRB time and is not zIIP eligible. The cost of archive log decryption also is charged as part of MSTR NONPREEMPT SRB time and is not zIIP eligible.

Table 3-27 provides a summary of where the CPU usage of data encryption and decryption of Db2 table spaces, index spaces, and active and archive logs is charged, and if the CPU cost is zIIP eligible or not. However, because of interconnectedness of everything within Db2, class 2 CPU, class 3 page latch suspension time, update commit time, and so on, might see variations too.

Table 3-27 Summary of data set encryption CPU time

Types of data set I/O operations	Main encryption CPU cost bearer	CPU cost accounted to	zIIP eligible?
DATABASE I/O	<i>ssnmDBM1</i>	IIT	No
PREFETCH	<i>ssnmDBM1</i>	IIT	No
DEFERRED WRITE	<i>ssnmDBM1</i>	PREEMPT SRB	Yes
CASTOUT	<i>ssnmDBM1</i>	PREEMPT SRB	Yes
ACTIVE LOG WRITE	<i>ssnmMSTR</i>	PREEMPT SRB	Yes
ACTIVE LOG READ	<i>ssnmMSTR</i>	IIT	No
ARCHIVE LOG WRITE	<i>ssnmMSTR</i>	NONPREEMPT SRB	No
ARCHIVE LOG READ	<i>ssnmMSTR</i>	NONPREEMPT SRB	No

### 3.12.3 Performance measurements for the REORG utility and encryption

Converting existing Db2 table spaces and indexes to use encrypted data sets requires a table space reorganization. We followed the conversion process and converted a table space with 1000 partitions to enable data set encryption and measured the **REORG** utility performance to understand the following performance aspects:

- ▶ How much of an effort it is to convert existing table spaces and indexes to use encrypted data sets.
- ▶ How the **REORG** utility performance is affected when using encrypted data sets.

#### Measurement environment

The following configuration was used for these measurements:

- ▶ IBM z14 processors.
- ▶ Two Crypto6S crypto cards that are configured the LPAR.
- ▶ 512 GB of LPAR storage.
- ▶ DS8870 disk controller.
- ▶ z/OS 2.3.
- ▶ Db2 12 FL 501 in a one-way data-sharing environment.

## Measurement scenario description

Table 3-28 describes the **REORG** measurements. The basic idea is to convert a table space with 1000 partitions and two indexes to use encrypted data sets and then maintain the objects with the data sets encrypted. We used the following three-step approach.

Table 3-28 *REORG TABLESPACE test scenarios*

Test ID	Db2 logs encrypted?	UNLDDS encrypted?	Pre-REORG table space encrypted?	Post-REORG table space encrypted?
Z14REONN	No	No	No	No
Z14REONE	Yes	Yes	No	Yes
Z14REOEE	Yes	Yes	Yes	Yes

1. Pre-conversion, as a baseline measurement: The **REORG** utility was run without any data sets being encrypted.
2. Conversion: Run the **REORG** utility after configuring the data sets with the proper encryption configuration (a RACF data set profile was used to assign the encryption key label). The Db2 log data sets also are encrypted.
3. Post-conversion: With the table space and its indexes using encrypted data sets, measure the **REORG** utility performance with all data sets encrypted.

The **REORG** utility tests were run when no other applications were accessing the table space. The **REORG** utility was run with following commands (we used the **LOG YES** option to stress the log activities):

```

TEMPLATE UNLDDS UNIT(SYSDA)
DSN(TPCEB0.DSNDBC.TPCEB100.&TS..P&PART.)
SPACE (3000,3000) CYL DISP(NEW,DELETE,CATLG)
LISTDEF REORGLST INCLUDE TABLESPACES
TABLESPACE TPCEB100.TEHHTS PARTLEVEL(1:1000)
REORG TABLESPACE LIST REORGLST UNLDDN UNLDDS
SORTDEVT 3390 SORTNUM 4 LOG YES PARALLEL 40
    
```

The **REORG** utility job was invoked with the **STATSLVL(SUBPROCESS)** parameter, and we used the utility statistics and OMPE reports to analyze the performance results.

## Measurement results

We observed the following results when we compared the conversion **REORG TABLESPACE** utility test (Z14REONE) to the pre-convention test (Z14REONN):

- ▶ The utility job runs approximately 4% longer. From the utility statistics, **UNLOAD**, **RELOAD**, and **SORTBLD** all had longer elapsed times. For more information, see Figure 3-77 and Figure 3-78.

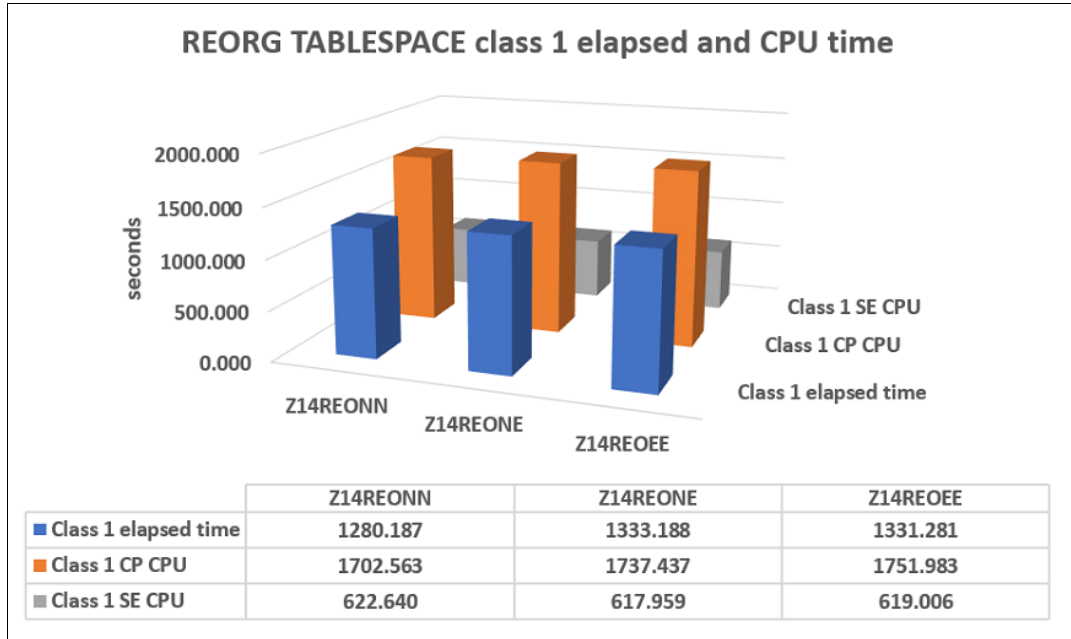


Figure 3-77 REORG TABLESPACE class 1 elapsed and CPU time

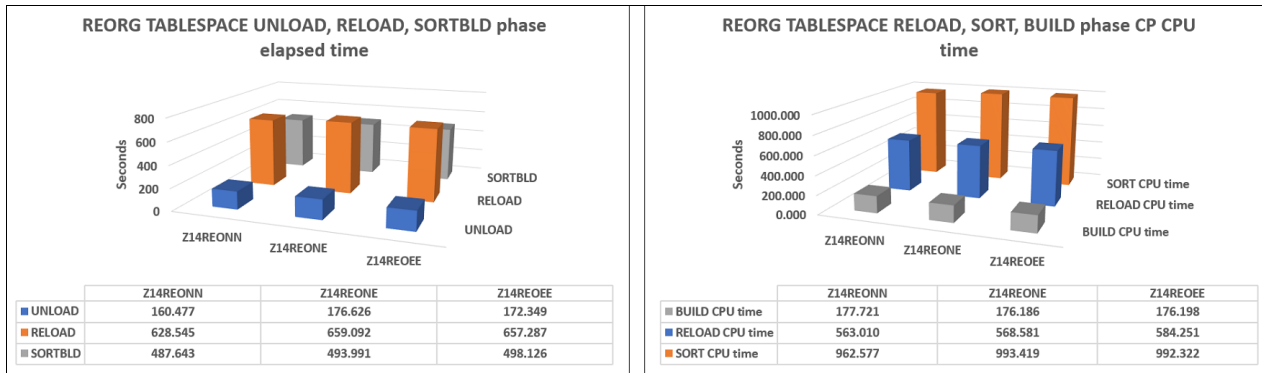


Figure 3-78 REORG TABLESPACE: Phase details

- ▶ The utility job uses 2% more class 1 CP CPU. Most of the CPU increase is from the **RELOAD** and **BUILD** phases, during which Db2 must read from the encrypted UNLDDS data sets. For more information, see Figure 3-77 and Figure 3-78.
- ▶ *ssnmDBM1* shows an increase of 5% in its IIP SRB time for writing data to encrypted table space and index data sets. For more information, see Figure 3-79 on page 141.

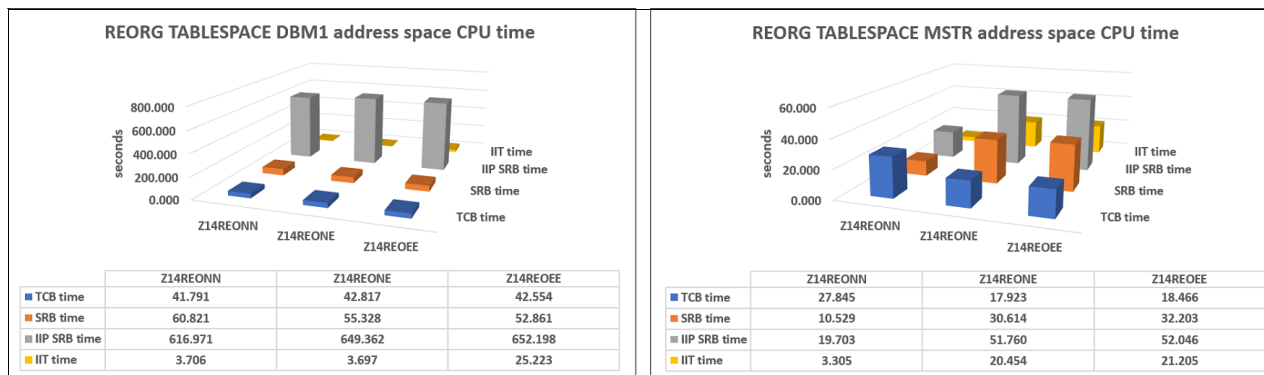


Figure 3-79 REORG TABLESPACE: Address space CPU time

- ▶ With both active logs and archive logs encrypted, the *ssnm*MSTR address space shows a 72% increase in total CPU usage. However, 76% of this CPU increase is used on a zIIP engine. For more information, see Figure 3-79.

We observed the following results when we compared the post-conversion **REORG TABLESPACE** utility test (Z14REOEE) to the pre-conversion test (Z14REONN):

- ▶ The utility job runs approximately 4% longer. From the utility statistics, **UNLOAD**, **RELOAD**, and **SORTBLD** all had longer elapsed times. For more information, see Figure 3-77 on page 140 and Figure 3-78 on page 140.
- ▶ The utility job uses 4% more class 1 CP CPU. Most of the CPU increase is from the **RELOAD** and **BUILD** phases, during which Db2 must read from encrypted UNLDDS data sets and write to encrypted table space and index data sets. For more information, see Figure 3-77 on page 140 and Figure 3-78 on page 140.
- ▶ *ssnm*DBM1 shows an increase of 6% in IIP SRB time for writing data into encrypted table space and index data sets. For more information, see Figure 3-79.
- ▶ With both active logs and archive logs encrypted, the *ssnm*MSTR address space shows a 77% increase in total CPU usage. However, 72% of this CPU increase is used on a zIIP engine. For more information, see Figure 3-79.

### 3.12.4 Performance measurements for GBP CF structure encryption

On Db2 systems running on z/OS 2.3 and later, Db2 data-sharing workloads can leverage the CF structure encryption feature and encrypt the data in the GBP and SCA structures.

For more information about setting up encrypted CF structures, see [Encrypting Coupling Facility structure data](#).

Like the z/OS data set encryption feature, the CF encryption feature also is an important part of the IBM Z pervasive encryption solution. To understand how Db2 workload performance can change by encrypting GBP structures, we designed and ran many workloads with heavy GBP CF structure read or write operations. This section describes the results of these tests.

Because SCA CF structure activity rates are usually low, we did not measure the SCA CF structure's encryption cost.

## Measurement environment

We used the following environment during this performance evaluation:

- ▶ IBM z14 processors: Eight general CPs and two zIIPs on each of the two hosting LPARs.
- ▶ Two Crypto6S crypto cards that are configured on the two hosting LPARs.
- ▶ LPAR storage configuration: 512 GB of storage for each of the two hosting LPARs.
- ▶ IBM DS8870 disk controller.
- ▶ Two ICFs, each with three CPs running with CFCC Level 22.
- ▶ z/OS 2.3.
- ▶ Db2 12 FL 502 in a two-way data-sharing environment.

## Measurement scenario description

This section describes the scenarios that we used to evaluate the performance that is associated with encrypting GBP CF structures.

Db2 GBP CF structure read activity is triggered by the following situations:

- ▶ Reading data because of a GBP-dependent getpage when the page is not found or was invalidated in the local buffer pool.
- ▶ Casting out data from GBP structures to disk that was triggered by GBP checkpoint operations or a cast-out threshold being hit.

Db2 writes to the GBP CF structures when the **GBPCACHE** option requires that the pages must be cached in GBP structure for GBP-dependent objects, which is either triggered by deferred writes or at transaction commit time.

To evaluate GBP prefetch read performance with CF encryption enabled, we used a full table space scan workload that reads data pages from the GBP through prefetch.

To evaluate how GBP read with CF encryption enabled performs without prefetch (which means that the GBP read requests are performed under the application thread), we used a semi-random select workload that reads one data page per select from the GBP.

We used a mass-sequential update workload on a GBP-dependent table space to trigger GBP write operations. The workload updates 2.1 million rows of data before committing. The GBP thresholds were set up in such a way that during the measurements no cast-out operations were triggered.

The table spaces that we used for these tests used a 4 K page size. Table 3-29 describes the workload characteristics.

*Table 3-29 GBP CF structure encryption tests workload characteristics*

GBP activity type	Workload used for evaluation	Number of GBP requests (thousands)
GBP READ for 4 K pages through prefetch	Full table space scan with data read from GBP	248.7
GBP READ for 4 K pages under application thread	Semi-random select with data that is read from GBP	1046.5
GBP WRITE for 4 K pages	Sequential update of GBP-dependent table space	212.9

## Measurement results

The tests are independent runs, each focusing on a different aspect of the CF encryption cost. The results that are presented in this section are the total CPU and elapsed time for each test.

The results of the table space scan tests, which involve reading and decrypting the GBP data from the CF structures through prefetch, are shown in Figure 3-80.

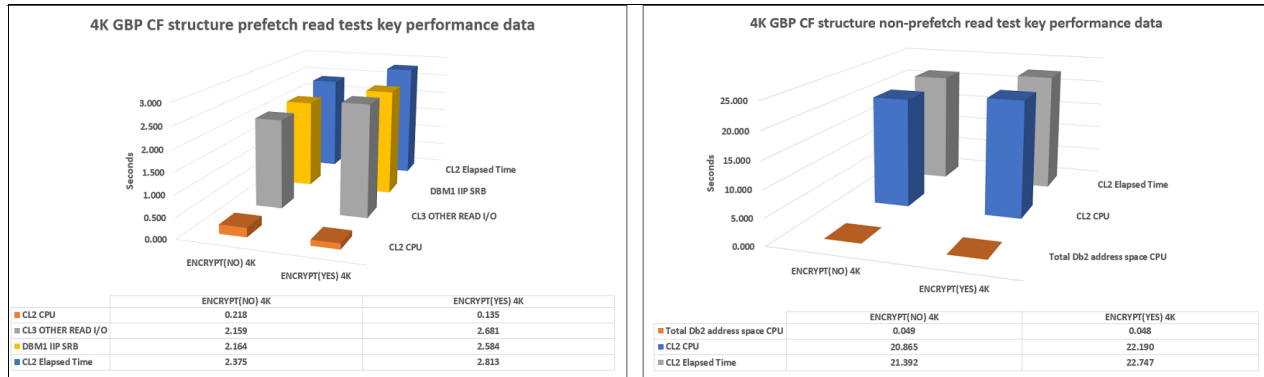


Figure 3-80 GBP structure encryption: Read operations

In summary, we made the following observations:

- ▶ Most of the CF structure decryption cost is charged to the *ssnmDBM1* address space. With approximately 95% of the data pages read from the GBP CF structure through prefetch, we observed a 19% increase of PREEMPT IIP SRB time for *ssnmDBM1*.
- ▶ Data decryption caused longer Other Read I/O time for prefetching the data pages from the encrypted GBP structures. As a result, when the GBP structures are encrypted, class 2 elapsed time might increase.

The results of the semi-random select tests, which involve reading and decrypting the GBP data from the CF structures under the application thread, are also shown in Figure 3-80. In summary, we made the following observations:

- ▶ Most of the CF structure decryption cost is charged to the class 2 CPU time of the application. With every data page read from the GBP CF structure, we observed a 6% increase in class 2 CPU time.
- ▶ The Db2 address spaces did not show a notable CPU usage difference.

The test results from the mass-sequential update measurements that used CF structure data encryption are shown in Figure 3-81.

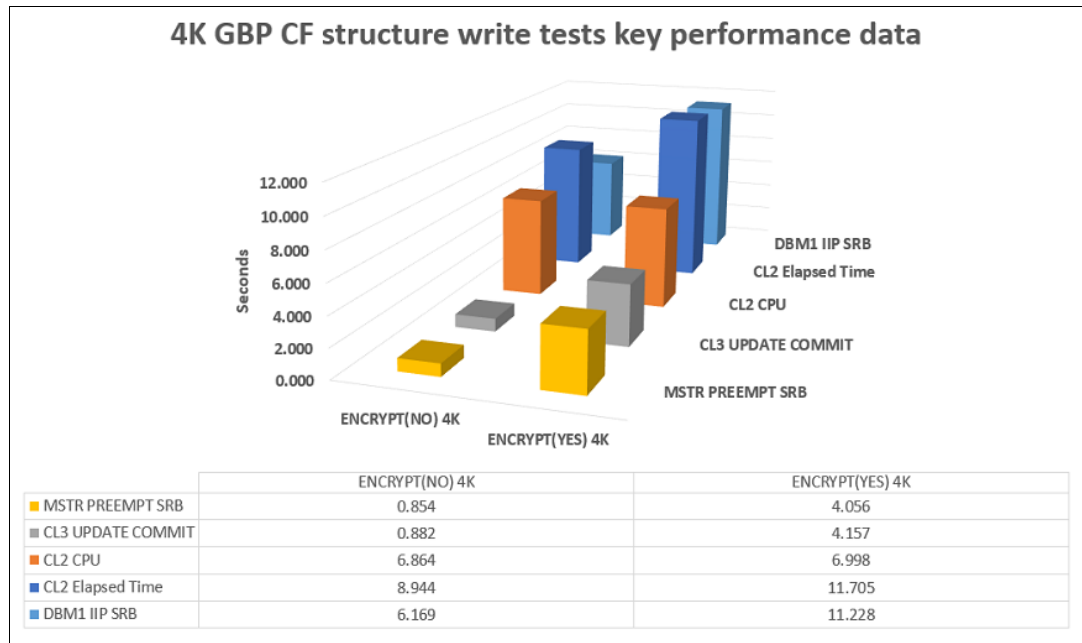


Figure 3-81 GBP structure encryption: Write operations

In summary, we made the following observations:

- ▶ The CF structure data encryption cost goes to two places:
  - For GBP synchronous writes, the *ssnm*MSTR address space spends non-zIIP eligible PREEMPT SRB time to do the encryption. For our test, we saw a 375% increase in *ssnm*MSTR PREEMPT SRB time.
  - For GBP asynchronous writes, the data encryption happens under the *ssnm*DBM1 address spaces, and we saw an increase of 82% in PREEMPT IIP SRB time.
- ▶ More data pages were synchronously written to the GBP during commit time because deferred writes must encrypt the data and become less efficient.
- ▶ Class 2 CPU time of the update application is similar, whether the GBP CF structure is encrypted or not.
- ▶ The update application had a longer update commit time because Db2 had to encrypt the data while writing to the GBP CF structure. The increased number of GBP synchronous writes also caused the update commit time to increase. As a result, class 2 elapsed time increased by 31%.

Table 3-30 on page 145 summarizes where the CPU usage is charged when encrypted GBP CF structures are used for the most common scenarios.



Table 3-30 GBP structure encryption CPU cost for read/write

GBP operations	Main CPU cost bearer	CPU cost accounted to	zIIP eligible?
GBP READs through prefetch	ssnmDBM1	PREEMPT SRB	Yes
GBP READs under the application thread	Application thread	Class 2 CPU	Maybe <sup>a</sup>
GBP READs by cast-out operations	ssnmDBM1	PREEMPT SRB	Yes
GBP ASYNCHRONOUS WRITE	ssnmDBM1	PREEMPT SRB	Yes
GBP SYNCHRONOUS WRITE	ssnmMSTR	PREEMPT SRB	No

a. Depending on the application thread type.

### 3.12.5 Performance measurements for IBM brokerage transaction workload

The specially designed workloads that we described in the previous sections helped us to understand how Db2 workload performance might be affected when running with z/OS data set encryption and CF structure encryption. However, they are not good real-life examples to determine how much overhead that you might see in your OLTP type workloads.

We used the IBM brokerage transaction workload to evaluate the performance impact of pervasive encryption on a real-life complex OLTP workload. This workload provides a better reference for users with OLTP workloads, which sets more appropriate expectations about the cost of running Db2 workloads with encrypted data sets and CF structures.

The IBM brokerage transaction workload runs various SQL/PL stored procedures, such as look up, update, insert, and reporting queries to simulate transactions in a brokerage firm. It accesses more than 17,000 objects (that is, tables and indexes), totaling 1.3 TB of data.

#### Measurement environment

The performance measurements were collected by using a two-way data-sharing configuration. Two types of CPUs, IBM z13 and IBM z14, were used to evaluate the encryption cost that is associated with each type of CPU. We used two environments to obtain our measurements:

- ▶ Environment 1:
  - Two IBM z13 LPARs with six dedicated general CPs each with a Crypto5S for secure key protection. Each LPAR has 512 GB of storage.
  - Two IBM z13 ICFs with three dedicated CPs and 256 GB storage each.
  - Four ICPs and four coupling over InfiniBand (CIB) links are shared between two LPARs and two ICFs.
  - z/OS 2.2 and CFCC Level 21 for the tests with only data sets encrypted.
  - When the data sets are encrypted, this situation is referred to as *Transparent Data Encryption (TDE)*, or *z/OS data set encryption*.
  - z/OS 2.3 and CFCC Level 21 for the tests with both data sets and CF structures encrypted.

- ▶ Environment 2:
  - Two IBM z14 LPARs with six dedicated general CPs, each with a Crypto6S for secure key protection. Each LPAR has 512 GB of storage.
  - Two IBM z14 ICFs with three dedicated CPs and 512 GB storage each.
  - Eight ICPs are shared between two LPARs and two ICFs.
  - z/OS 2.2 and CFCC Level 21 for the tests with only data sets encrypted.
  - z/OS 2.3 and CFCC Level 21 for the tests with both data sets and CF structures encrypted.

### **Measurement scenario description**

The encrypted IBM brokerage transaction workload includes the following elements:

- ▶ Db2 active and archive logs are encrypted.
- ▶ The Db2 directory and catalog are encrypted.
- ▶ Most table spaces and indexes are encrypted, which results in five out of eight sync I/Os per transaction against encrypted objects with a 4 K page size.
- ▶ All GBPs and SCA structures are encrypted during CF encryption measurements.

### **Measurement results**

During the measurement period, transactions run at a rate of approximately 5,000 per second for the data-sharing group with two members. In terms of the amount of work that is done by encryption and decryption, it translates into the following metrics:

- ▶ 25,000 4 K pages were decrypted per second. This metric is the result of five sync I/Os per transaction against encrypted objects with a 4 K page size.
- ▶ 10,000 4 K pages were decrypted per second. This metric is the result of two cast-out pages per transaction against encrypted objects with a 4 K page size.
- ▶ 7,000 encrypted CIs per second were written to active log data sets by two members of the data-sharing group.
- ▶ 10,000 CIs per second were decrypted and encrypted through the archive log process by reading from encrypted active log data sets and writing to encrypted archive log data sets by two members.
- ▶ There were no I/Os to the Db2 directory or catalog during the measurement interval. Therefore, no encryption or decryption work is needed.
- ▶ Encrypting and decrypting 70,000 4 K pages per second was done by CF encryption and decryption for all GBPs and SCA structures.

Overall, the rate of encryption and decryption by data set encryption alone for this workload was approximately 52,000 4 K pages per second (a rate of 122,000 4 K pages per second when CF encryption was enabled).

In Figure 3-82 on page 147 and Figure 3-83 on page 147, the “Base” test is the measurement that is run without any data set or CF structures that are encrypted. The “TDE” test is the measurement with only data sets, including all active and archive logs, and the selected table space and index data sets are encrypted. The “TDE with CF” test is the measurement that is done with the same data sets encrypted as the “TDE” test, and with all the GBP and SCA structures encrypted.

On IBM z13 processors, the encryption cost for data set encryption, in terms of ITR loss, is observed at 2.3%, but the ITR loss is at 3.8% when both TDE and CF encryption are in effect. Minimal differences in transaction elapsed time were observed. Figure 3-82 illustrates the encryption cost, in terms of ITR loss, for this workload running on an IBM z13 processor.

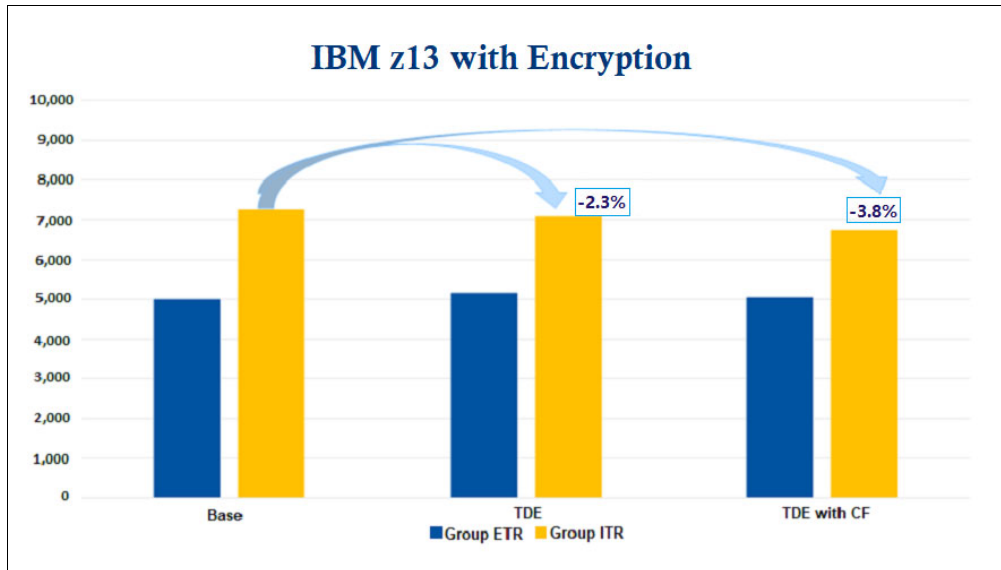


Figure 3-82 IBM brokerage transaction workload: encryption on IBM z13

On z14 processors, the encryption cost for TDE, in terms of ITR, is observed at 0.4%, but the ITR loss is at 2.4% when both TDE and CF encryption are in effect. Again, minimal differences in transaction elapsed time were observed. Figure 3-83 illustrates the encryption cost, in terms of ITR loss, for this workload running on an IBM z14 processor.

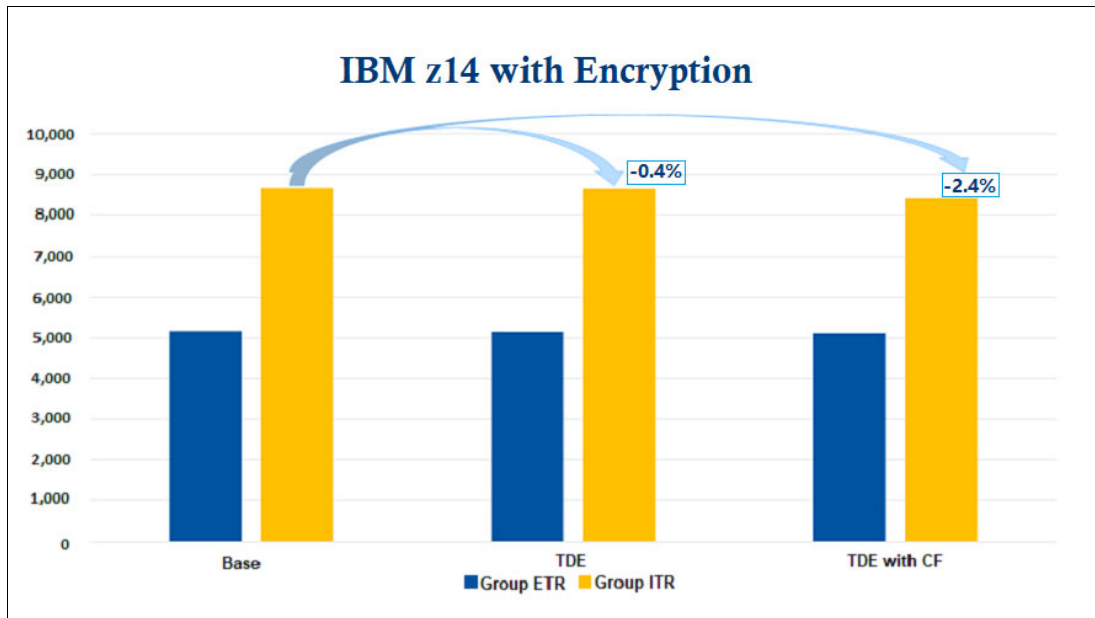


Figure 3-83 IBM brokerage transaction workload: encryption on IBM z14

### 3.12.6 Monitoring information

Db2 does not provide specific traces to record data-set-level encryption statistics and performance. The encryption feature is transparent to users. You can check SMF type 42 subtype 6 records to get information about how much data is being encrypted and decrypted for encrypted data sets.

CF structure encryption is transparent to Db2, and Db2 does not provide extra traces to track GBP or SCA structure encryption statuses and performance.

The **-DISPLAY GROUP**, **-DISPLAY LOG**, and **-DISPLAY ARCHIVE** commands can be used to display the encryption key labels that the Db2 data-sharing group, the current active log data sets, and archive log data sets are using when you are running with Db2 12 at FL 502 and later.

### 3.12.7 Usage considerations

For more information about how to set up data set encryption for various types of Db2 objects, see [Encrypting your data with z/OS DFSMS data set encryption](#).

Protecting data comes with a price. Before implementing data set encryption and CF structure encryption, you can use the no-charge zBNA tool that is provided by IBM to estimate the cost of encryption based on your current workload characteristics. An IBM Community article, [zBNA for Pervasive Encryption Estimation](#), describes how the tool is used to estimate encryption cost for data sets and CF structures. To download the tool, see [IBM Z Batch Network Analyzer \(zBNA\) Tool](#).

If your Db2 subsystems have data that is sensitive enough that it needs more protection, with all the information that is provided in this section, you can start planning to implement data set encryption and CF structure encryption with a data-set-by-data-set and structure-by-structure approach.

### 3.12.8 Conclusion

When running Db2 workloads with encrypted data sets and CF structures, there are some CPU and elapsed time costs. The CPU cost mostly goes to Db2 address spaces, that is, *ssnmMSTR* and *ssnmDBM1*. Class 2 CPU time usually is not impacted. Class 2 elapsed time can see some degree of increase because each I/O operation and CF data read/write operation needs more time to encrypt or decrypt data.

Our in-lab tests show that CPU and elapsed time costs are lower when running with a newer IBM zSystems processor (IBM z14 or later) that includes a pervasive encryption solution, rather than on older IBM zSystems processor models (IBM z13 or earlier).



## Workload-level measurements

This chapter describes the results of some key performance measurements by using different IBM internal Db2 for z/OS workloads.

The following workloads were used to evaluate the overall performance of Db2 13 for z/OS compared to Db2 12 for z/OS:

- ▶ Performance regression bucket
- ▶ IBM brokerage transaction workload (version 1)
- ▶ IBM brokerage transaction workload (version 2): Effect of REBIND (one-way data sharing) on performance
- ▶ Distributed IBM Relational Warehouse Workload workload
- ▶ High-insert batch workloads
- ▶ Relational transaction workload (CICS and Db2)
- ▶ IBM query regression workload
- ▶ SAP banking workloads

For more information about the workloads that were used, see Appendix A, “IBM Db2 workloads” on page 403.

## 4.1 Performance regression bucket

The performance regression bucket (PRB) is a comprehensive batch benchmark that is created internally for daily Db2 performance regression measurements. It consists of nine sub-workloads that cover the major Db2 for z/OS performance areas, such as:

- ▶ Data definition statements (Data Definition Language (DDL))
- ▶ Bind
- ▶ Query
- ▶ Column **INSERT** and **FETCH**
- ▶ Batch (insert, update, delete, select, fetch, merge, overflow, pagination, archive transparency, and basic and advanced triggers)
- ▶ Utilities (**LOAD**, **UNLOAD**, **COPY**, **RUNSTATS**, **REBUILD**, **REORG**, and **RECOVER**)
- ▶ Distributed Online Transaction Processing (OLTP), large object (LOB), and XML
- ▶ Data sharing, concurrency, encryption, compression, plan stability, and so on

### 4.1.1 Performance measurement description

Performance measurements were conducted by using a two-way data-sharing environment on the following environment:

- ▶ One IBM z14 logical partition (LPAR) with six general CPs and three IBM zSystems Integrated Information Processors (zIIPs) running both Db2 data-sharing members
- ▶ z/OS 2.4
- ▶ 56 GB of real storage

Db2 13 performance is compared to Db2 12 performance by using default Db2 subsystem parameter settings:

- ▶ Db2 12 baseline performance measurements were done at function level (FL) V12R1M510 and catalog level V12R1M509.
- ▶ Db2 13 performance measurements were done at FL V13R1M501 and catalog level V13R1M501.

### 4.1.2 Performance results

On average for the entire PRB workload, Db2 13 has about 1.7% CPU time improvement compared to Db2 12.

As illustrated in Figure 4-1 on page 151, the following results were measured at the sub-workload level:

- ▶ For DDL, BIND, column, query, XML, and LOB workloads, Db2 13 is almost equivalent to Db2 12.

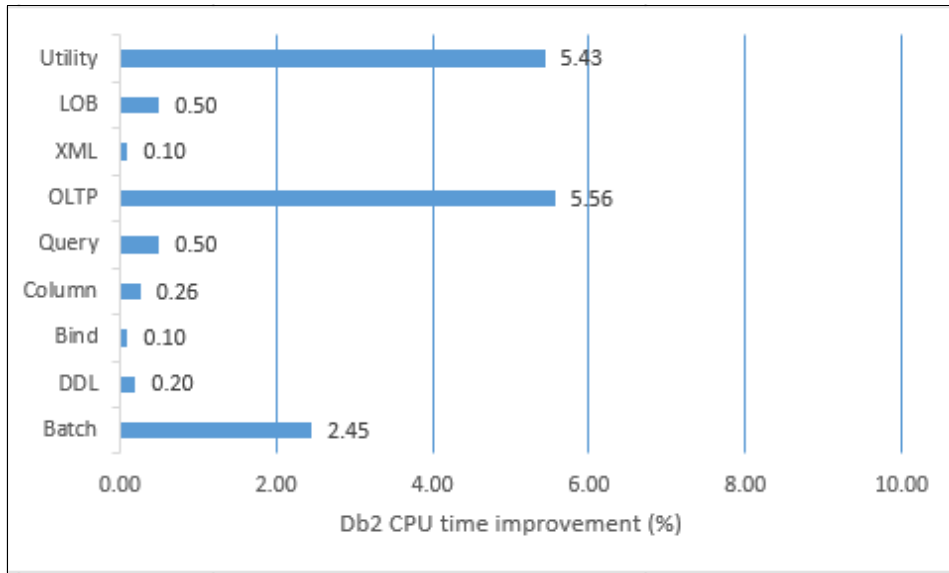


Figure 4-1 CPU time improvement for PRB sub-workloads for Db2 12 versus Db2 13

- For batch workloads, on average Db2 13 has about 2.5% CPU time improvement compared to Db2 12. Inside the batch sub-workload, the CPU time improvement for individual batch jobs can be up to 20% (see Figure 4-2). The improvement is mostly due to the index buffer pool getpage reduction, which results from the index look-aside optimization for sequential insert, update, and delete processing or the fast index traversal enhancements, including fast traversal blocks (FTB) enablement for non-unique indexes by default in Db2 13, and the index key size increase for FTB eligibility. For more information, see 2.6, “Fast index traversal enhancements” on page 27.

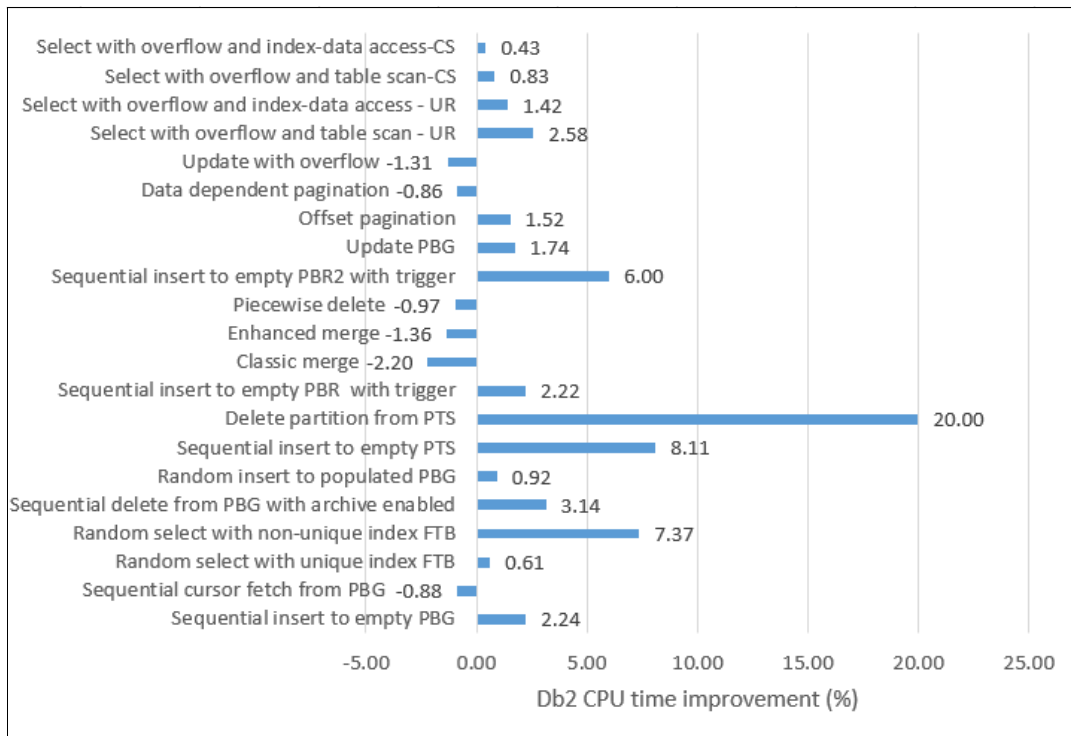


Figure 4-2 CPU time improvement for batch jobs for Db2 12 versus Db2 13

- ▶ When comparing the OLTP sub-workload, Db2 13 shows about 5.5% CPU time improvement compared to Db2 12. The CPU time improvement mainly comes from the index look-aside enhancement for sequential insert, update, and delete processing, and the fast index traversal enhancements, including FTB enablement for non-unique indexes by default, and the FTB key size limit increase from 64 bytes to 128 bytes.
- ▶ For the utility sub-workload, on average Db2 13 has about 5.4% CPU time improvement compared to Db2 12. For the **LOAD**, **UNLOAD**, **REBUILD INDEX**, **RUNSTATS**, **COPY**, and **RECOVER** utilities, Db2 13 is almost equivalent to Db2 12 (see Figure 4-3).

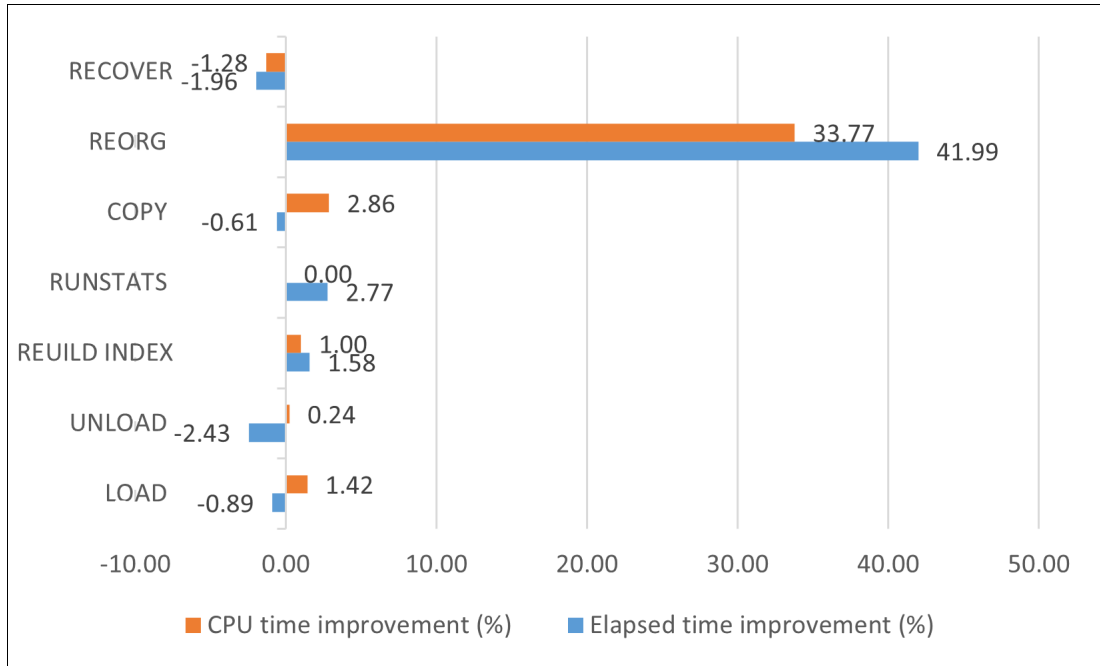


Figure 4-3 Elapsed and CPU time improvement for the utility subworkload for Db2 12 versus Db2 13

A large improvement was seen in the set of **REORG** utility test cases, as shown in Figure 4-3. This improvement is due to improvements in **REORG INDEX** performance, as shown in Table 4-1. This performance improvement is the result of using the **NOSYSUT1** keyword for **REORG INDEX** as the default system parameter. The **NOSYSUT1** enhancement (described in more detail in 9.7, “REORG INDEX NOSYSUT1” on page 303) avoids the usage of an **SYSUT1** data set (**NOSYSUT1**), and it enables utility subtasks to unload and rebuild the index keys in parallel.

Table 4-1 Elapsed and CPU time improvement for REORG for Db2 12 versus Db2 13

Parameter	Elapsed time improvement (%)	CPU time improvement (%)
REORG TABLESPACE	-0.73	0.12
REORG INDEX	84.70	67.41

Real storage usage for the DBM1 address space increased by about 400 MB (about 2%) in Db2 13 compared to Db2 12. The storage increase comes from more FTB usage and the **REORG INDEX NOSYSUT1** performance improvement.



### 4.1.3 Conclusion

When migrating from Db2 12 to Db2 13, you can expect some CPU time improvement in sequential insert, update, delete, random data access, and OLTP workloads, which benefit from sequential index look-aside optimization and the FTB enhancements. You also can expect a significant elapsed and CPU time improvement for **REORG INDEX** due to the **REORG INDEX NOSYSUT1** performance enhancement.

## 4.2 IBM brokerage transaction workload (version 1)

The IBM brokerage transaction workload is a complex OLTP workload that runs various SQL stored procedures to simulate transactions in a brokerage firm. In Version 2 of this workload, broker volume and data maintenance transactions are removed from the transaction mix to better maintain scalability from non-data-sharing measurements to one-way and two-way data-sharing measurements.

### 4.2.1 Two-way and one-way data-sharing and non-data-sharing performance measurements

This workload is migrated from Db2 12 at FL 510 to Db2 13 at FL 100. Next, Db2 13 FLs 500 and 501 are activated. Finally, all applications are rebound with the **APREUSE(ERROR)** option to preserve the access paths from Db2 12 at FL 510.

Performance measurements were captured for both Db2 12 FL 510, which serves as the baseline, and Db2 13 FL 501 to demonstrate performance enhancements.

### 4.2.2 Performance measurement

The measurements were performed by using the following environment:

- ▶ A sysplex running on an IBM z15 machine, including two LPARs and two Coupling Facilities (CFs):
  - Three CPs and three zIIPs on each LPAR, MT=2, and 512 GB on each LPAR running on z/OS 2.5.
  - Two internal CFs with three dedicated CPs and 256 GB of memory on each CF, running CFLEVEL 24.
  - CS5 coupling links are employed.

Table 4-2 shows the results for the two-way data-sharing measurements.

Table 4-2 Db2 CPU time per transaction enhancement and getpage reduction for two-way data sharing

Db2 version	Db2 12 (FL 510)	Db2 12 (FL 510)	Db2 13 (FL 501)	Db2 13 (FL 501)	Delta
Db2 member	MEM1	MEM2	MEM1	MEM2	
<b>Central processor (CP) and SE CPU Time (ms/commit)</b>					
Class 2	0.852	0.874	0.843	0.858	1.4%
MSTR	0.009	0.009	0.010	0.010	
DBM1	0.014	0.023	0.014	0.025	
Internal resource lock manager (IRLM)	0.028	0.029	0.031	0.032	
Total Db2 CPU Time (Class 2 + MSTR + DBM1 + IRLM) (member scope)	0.903	0.935	0.898	0.925	
Total Db2 CPU Time (group scope)	0.919		0.912		-0.8%
GetPage/Commit	241.86	241.95	125.41	125.59	-48.1%

Table 4-3 shows the results for the one-way data-sharing measurements.

Table 4-3 Db2 CPU time per transaction enhancement and getpage reduction for one-way data sharing

Db2 version	Db2 12 (FL 510)	Db2 13 (FL 501)	Delta
<b>CP and SE CPU Time (ms/commit)</b>			
Class 2	0.644	0.611	-5.1%
MSTR	0.005	0.005	
DBM1	0.006	0.006	
IRLM	0.001	0.001	
Total Db2 CPU Time (Class 2 + MSTR + DBM1 + IRLM)	0.656	0.623	-5.0%
GetPage/Commit	241.89	121.36	-49.8%

Table 4-4 shows the results for the measurements in a non-data-sharing environment.

Table 4-4 Db2 CPU time per transaction enhancement and getpage reduction for non-data sharing

Db2 version	Db2 12 FL 510	Db2 13 FL 501	Delta
<b>CP and SE CPU Time (ms/commit)</b>			
Class 2	0.613	0.595	-2.9%
MSTR	0.005	0.005	
DBM1	0.006	0.006	
IRLM	0.000	0.000	
Total Db2 CPU Time (Class 2 + MSTR + DBM1 + IRLM)	0.624	0.606	-2.9%
GetPage/Commit	241.85	121.44	-49.8%

Table 4-5 shows the effect of the increased number of FTB eligible indexes on the notify traffic, which results in higher IRLM CPU usage.

Table 4-5 Increased traffic in Notify Message with more FTB eligible objects

Db2 version	Db2 12 (FL 510)	Db2 12 (FL 510)	Db2 13 (FL 501)	Db2 13 (FL 501)	Delta
Db2 member	MEM1	MEM2	MEM1	MEM2	
Number of FTB Objects	1082	1077	1806	1783	+66%
<b>Notify messages (/commit)</b>					
Sent	0.01	0.00	0.04	0.04	+800%
Received	0.00	0.01	0.04	0.04	+800%
IRLM (ms/commit)	0.028	0.029	0.031	0.032	+10%

Many internal optimizations and enhancements, which apply to Db2 13 at FL 501 and are not applicable to Db2 12, contribute to performance improvement. However, fast index traversal, also known as FTB, is the one of the most important contributors to performance enhancement for this workload in Db2 13 at FL 501. For more information, see 2.6, “Fast index traversal enhancements” on page 27.

Table 4-2 on page 154, Table 4-3 on page 154, and Table 4-4 show a reduction in getpage count per commit in Db2 13 at FL 501, compared to Db2 12 FL 510: a reduction of 48.1% for two-way, 49.8% for one-way and non-data sharing respectively. This reduction translates into a reduction in transaction class 2 CPU time, which results in an enhanced workload performance of 0.8% for two-way data sharing, 5.0% for one-way data sharing, and 2.9% in a non-data-sharing environment.

As shown in Table 4-5 on page 155, the number of objects that use FTB is higher in Db2 13 at FL 501 than in Db2 12 at FL 501 because of the expanded key length for eligible indexes and the enablement of the `FTB_NON_UNIQUE_INDEX` subsystem parameter by default. The more FTB objects that exist, the higher the likelihood that the FTB structure is modified, which drives more notify messages to other Db2 data-sharing members. IRLM is the workhorse to deliver these messages, so IRLM incurs higher CPU time, as shown in Table 4-5 on page 155. Nonetheless, the increased IRLM CPU time is offset by the improved Db2 class 2 CPU time, as shown in Table 4-2 on page 154. Overall, the combined CPU improvement is the most important arbitrator.

The RMF XCF activity report is monitored to ensure the wellness of the sysplex signaling situation even with the increased notify traffic.

The Db2 13 FL 501 DBM1 storage footprint is expected to increase because more index objects are eligible to use FTB than in Db2 12. Monitoring of this workload determined that FTB storage usage is about 1.5 GB more per Db2 member in Db2 13 FL 501 than in Db2 12. Specifically, with Db2 12 FL 510, FTB total memory allocation is 427 MB out of 5070 MB of DBM1 real storage usage, excluding a virtual buffer pool. With Db2 13 at FL 501, the FTB total memory allocation is 1900 MB out of 7120 MB of DBM1 real storage usage other than a virtual buffer pool. The PTF for APAR PH46301, which addresses an excessive DBM1 storage usage issue, is applied in the measurement.

### 4.2.3 Conclusion

In summary, migration of the IBM brokerage transaction workload from Db2 12 at FL 510 to Db2 13 FL 501 in two-way data-sharing, one-way data-sharing, and non-data-sharing environments, shows performance improvement, as indicated by a decrease in Db2 CPU time per transaction. DBM1 real storage usage in Db2 13 at FL 501 increased compared to Db2 12, due to the expanded FTB usage in Db2 13.

## 4.3 IBM brokerage transaction workload (version 2): Effect of REBIND (one-way data sharing) on performance

The IBM brokerage transaction workload is a complex OLTP workload that runs various SQL stored procedures to simulate transactions in a brokerage firm. This version of IBM brokerage transaction workload contains the full set of a transaction mix. Broker volume and data maintenance transactions are not removed from the transaction mix during these measurements. For more information about the workload, see Appendix A, “IBM Db2 workloads” on page 403.

### 4.3.1 Requirements

The following levels of Db2 are required for running this workload:

- ▶ A Db2 13 FL 100 subsystem
- ▶ A Db2 12 FL 510 subsystem to compare against

## 4.3.2 Performance measurements

For this performance study, the IBM brokerage transaction workload in a one-way data-sharing configuration is migrated from Db2 12 at FL 510 to Db2 13 at FL 501. Performance measurements are conducted at each level of the migration to track how the workload performs at each step of the migration process. Separate measurements also are conducted both before and after rebinding the packages that the workload uses.

### Measurement environment

All performance measurements are performed by using the following environment:

- ▶ IBM z15 LPAR with eight general CPs
- ▶ z/OS 2.5

### Measurement scenario description

The IBM brokerage transaction workload was run with Db2 13 at the following levels, and the results were compared to Db2 12 at FL 510:

- ▶ Db2 13 at FL 100 without rebinding any packages.
- ▶ Db2 13 at FL 100 after rebinding all packages with **APREUSE(ERROR)** to ensure that the same access paths are used in Db2 13.
- ▶ Db2 13 at FL 500 after rebinding all packages with **APREUSE(ERROR)** to ensure that the same access paths are used in Db2 13.
- ▶ Db2 13 at FL 501 after rebinding all packages with **APREUSE(ERROR)** to ensure that the same access paths are used in Db2 13.

The Db2 FL 510 one-way data-sharing measurement is used as the baseline.

## 4.3.3 Performance results

The IBM brokerage transaction workload runs 30 stored procedures and achieves a transaction rate of 6800 - 6900 transactions per second.

Figure 4-4 shows the CPU improvement for IBM brokerage workload (one-way data sharing) at each Db2 13 migration step. The Db2 class 2 CPU time improvement percentage that was observed at different steps of the migration process also is shown.

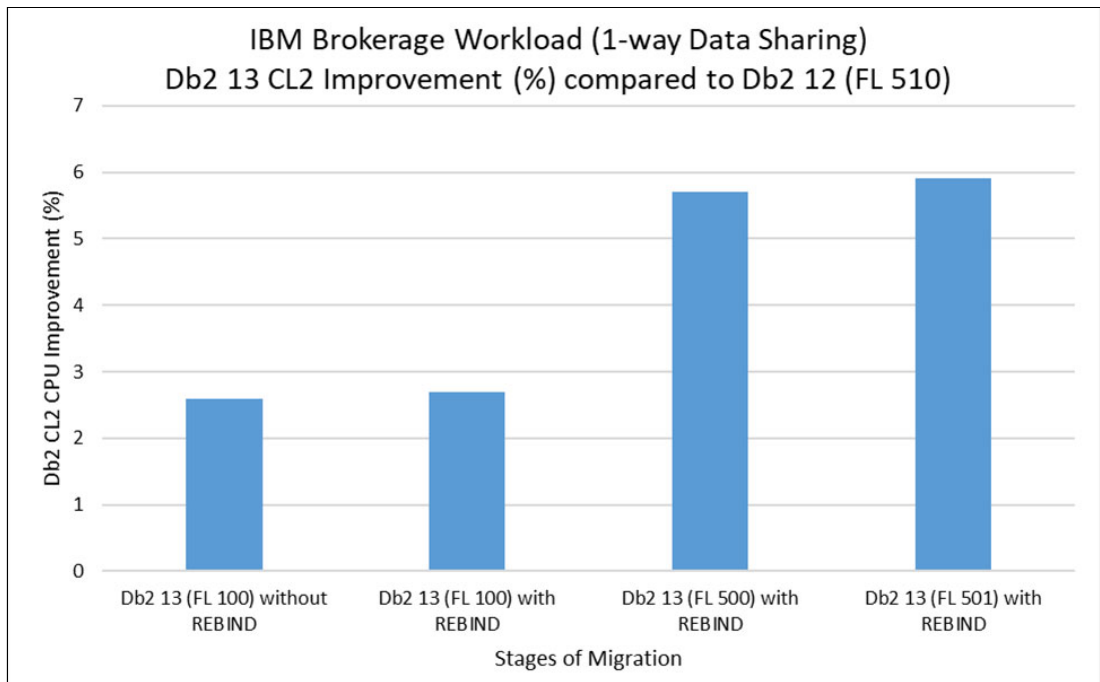


Figure 4-4 CPU improvement for IBM brokerage workload (one-way data sharing)

After migration to Db2 13 FL 100 without running a **REBIND** of the stored procedure packages, the CPU time is reduced by 2.6% compared to the Db2 12 baseline test.

Then, a **REBIND** is run for all packages specifying **APREUSE(ERROR)** to ensure that the same access paths are used for all the packages. In this scenario, CPU time is reduced by 2.7% compared to the Db2 12 baseline.

The next measurement is performed after activating Db2 13 FL 500 and a **REBIND** by using **APREUSE(ERROR)** for all the packages. A further improvement for a Db2 class 2 CPU time of 5.7% was observed compared to the Db2 12 baseline.

The final measurement is performed after activating Db2 13 FL 501 and a **REBIND** by using **APREUSE(ERROR)** for all packages. This measurement resulted in a further slight improvement of 5.9% Db2 class 2 CPU time compared to the Db2 12 baseline.

Most of the reduction in Db2 class 2 CPU time for IBM brokerage transaction workload in Db2 13 can be attributed to the reduction in buffer pool getpage from more indexes being eligible for in-memory fast index traversal.

In Db2 13, the default parameter setting of **FTB\_NON\_UNIQUE\_INDEX** is changed from **N0** to **YES**, thus making more indexes eligible for in-memory fast index traversal.

Furthermore, in Db2 13 at FL 500, the key size of eligible unique indexes was increased from 64 bytes to 128 bytes, and the key size of eligible non-unique indexes was increased from 56 bytes to 120 bytes. With more indexes eligible for in-memory fast index traversal, fewer buffer pool getpages are needed for random index access, as shown in Figure 4-5 on page 159. The figure shows the buffer pool getpages for the IBM brokerage workload (one-way data sharing) at each Db2 13 migration step.

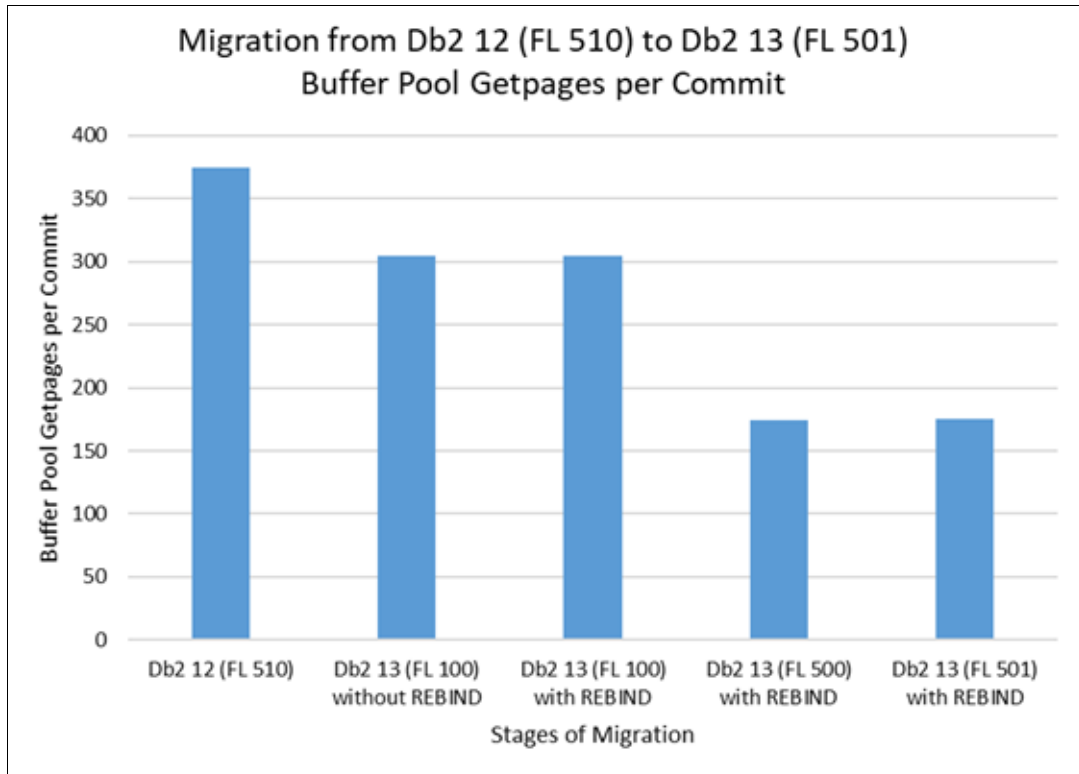


Figure 4-5 Buffer pool getpages for the IBM brokerage workload (one-way data sharing)

After migration from Db2 12 to Db2 13 at FL 100, the buffer pool getpages per commit dropped from 375 to 305. Then, after migrating to Db2 13 and activating FL 500, buffer pool getpages per commit further dropped further to 174.

Figure 4-6 shows the total real storage usage by the IBM brokerage workload (one-way data sharing) at each Db2 13 migration step. With more indexes eligible for in-memory fast index traversal in Db2 13, we also observed an increase in real storage usage during different stages of migration, as shown in Figure 4-6.

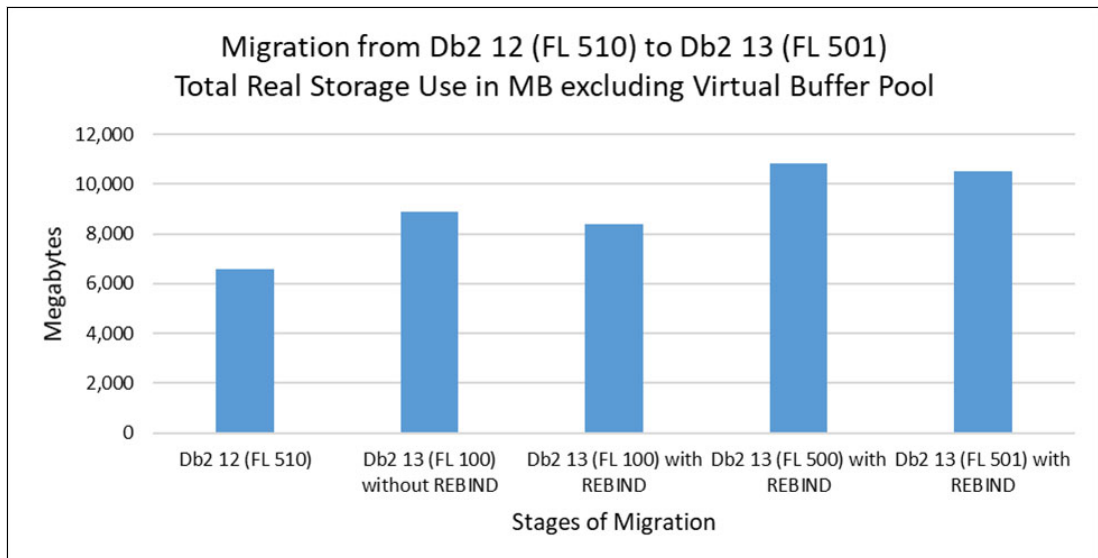


Figure 4-6 Total real storage usage by the IBM brokerage workload (one-way data sharing)

When migrating from Db2 12 to Db2 13 at FL 100, total real storage increased by about 2,300 MB compared to Db2 12. The increase occurs because more indexes become eligible for in-memory fast index traversal due to the default setting change for the **FTB\_NON\_UNIQUE\_INDEX** subsystem parameter to YES.

Then, after Db2 13 FL 500 was activated, total real storage usage increased to about 4,200 MB compared to Db2 12. The further increase in total real storage occurs because more indexes become eligible for in-memory fast index traversal due to the increased key size for eligible unique and non-unique indexes.

### 4.3.4 Conclusion

After migrating to Db2 13 FL 100 without rebinding your packages, you can expect a reduction in CPU usage if non-unique indexes are eligible to use fast index traversal. Starting at Db2 13 FL 500, you might see further reduction in CPU as more indexes become eligible for fast index traversal. As more indexes qualify for fast index traversal, a reduction in buffer pool getpage occurs, which amounts to less CPU usage. You can expect real storage usage to grow as more indexes use in-memory fast index traversal.

## 4.4 Distributed IBM Relational Warehouse Workload workload

To evaluate the performance of Db2 13 in a distributed environment, the distributed IBM Relational Warehouse Workload (IRWW) workload suite was used. This workload suite includes different implementations:

- ▶ **SQLCL**: A CLI (dynamic) that uses the Db2 CLI driver.
- ▶ **JDBC**: A JDBC (dynamic) that uses the JCC T4 driver.
- ▶ **JDBCT2**: A JDBC dynamic workload that uses the JCC T2 driver.
- ▶ **SQLJ**: A static Java SQLJ application that uses the SQLJ driver.
- ▶ **SPNS**: Stored procedures on native SQL (static) that use the JCC T4 driver.
- ▶ **SPSJ**: Stored procedures in SQLJ (static) that use the JCC T4 driver.
- ▶ **SPCB**: Stored procedures in COBOL (static) that use the JCC T4 driver.

All workloads are multiple thread applications and consist of seven transactions. Each transaction runs one or multiple SQL statements to perform business functions in a predefined mix, as shown in Table 4-6.

*Table 4-6 IBM Relational Warehouse Workload transaction mix*

Transaction	% mix	Data Manipulation Language (DML) operations
Delivery	2	OPEN, FETCH, SELECT, UPDATE, and DELETE
Now Order	22	OPEN, FETCH, SELECT, INSERT, and UPDATE
Order Status	24	OPEN FETCH, and SELECT
Payment	22	OPEN, FETCH, SELECT, INSERT, and UPDATE
Price Change	1	UPDATE
Price Quote	25	SELECT
Stock Level	4	SELECT



The distributed workloads run in an environment with the following configuration:

- ▶ IBM z15 LPAR that uses z/OS 2.5 with three dedicated general CPs, two dedicated zIIPs, and 128 GB of storage.
- ▶ Linux on IBM Z on an IBM z15 LPAR with six CPs and 24 GB of memory.
- ▶ IBM Data Server Driver for JDBC and SQLJ 4.31.10 for T4 driver, and IBM Data Server Driver for JDBC and SQLJ 4.31.11 for T2 driver.
- ▶ IBM Java virtual machine (JVM) 9 (build 2.8, JRE 1.8.0 Linux s390x-64 20170722\_357405 (JIT enabled, accelerator-only table (AOT) enabled)).
- ▶ HiperSockets communication between the Linux on IBM Z and the z/OS LPARs.
- ▶ All packages are bound by using the **RELEASE(DEALLOCATE) BIND** option.

Total transaction CPU time per commit and internal throughput rate (ITR) are the two main metrics that are used for comparing transactions running on Db2 13 against Db2 12.

#### 4.4.1 Distributed IBM Relational Warehouse Workload CPU and ITR comparison

The measurement data was collected on the Db2 server side. From the collected data, Db2 accounting, statistics, and RMF reports were generated for analysis. For non-stored-procedure workloads and the native stored procedure workload (SPNS), the total CPU time per commit serves as the main measure of the CPU cost during these performance tests. It is calculated by adding up the total address space CPU time per commit from each Db2 subsystem address space (MSTR, DBM1, DIST, and IRLM) in the Db2 statistics report. This total CPU time includes the CPU cost of the DDF communication, Db2 agent CPU cost, and Db2 engine cost. For other stored procedure workloads, the total transaction CPU is the statistics total address space CPU per commit, plus accounting class 1 STORED PRC CPU in the IBM OMEGAMON for Db2 Performance Expert (OMPE) report.

The values in Table 4-7 represent the percentage difference between the measurements on Db2 12 FL 510 and Db2 13 FL 501.

Table 4-7 Distributed workloads: Delta CPU% and ITR% with different PKGREL options in effect

Workload	PKGREL(COMMIT)		PKGREL(BNDOPT)	
	Total CPU Time per commit	ITR	Total CPU Time per commit	ITR
SQCL	-3.11	3.01	-6.67	0.56
JDBC	-2.41	8.22	-14.15	2.25
SQLJ	-0.41	0.27	-0.5	1.09
SPNS	-3.16	0.48	-1.60	2.73
SPCB	-1.39	0.19	-6.67	0.56
SPSJ	-5.80	6.75	-10.00	9.75
JDBCT2	-18.53	5.75	-19.22	7.11

All distributed workloads show a CPU cost reduction. Depending on the workload, the reduction can be up to 3% with **PKGREL (COMMIT)**, and 14% with **PKGREL (BNDOPT)**. The ITR also increased, up to 8% with **PKGREL (COMMIT)** and 2.74% with **PKGREL (BNDOPT)**. A number of optimizations of how the IBM Data Server Driver for JDBC and SQLJ handles “wrapper classes” lead to a larger CPU reduction in Db2 13 for this workload, which is an indicator that Db2 13 is slightly better compared to Db2 12. The improvements are the result of enhancements in the fast index traversal feature, storage management optimization, and other areas of improvement in the DDF area and in the Db2 engine in Db2 13.

## 4.5 High-insert batch workloads

The high insert batch workload (HIW) is a set of complex workloads that are related to applications with high insert rates for the following batch workload scenarios:

- ▶ Archive sequential insert
- ▶ Journal table insert
- ▶ Message queues insert and delete
- ▶ Random insert and delete
- ▶ Massive simple rows insert

**Note:** Unless indicated otherwise, the elapsed and CPU times in the figures in this section are in *seconds*.

### 4.5.1 Performance measurement environment

All performance measurements were performed by using the following environment:

- ▶ IBM z15 LPAR with eight general CPs and no zIIPs
- ▶ z/OS 2.4
- ▶ Db2 13 for z/OS at FL 501
- ▶ Db2 12 at FL 503 (used for the baseline measurements)

### 4.5.2 Performance results

This section describes the results for each of the batch workload scenarios.

#### Archive sequential insert

In this workload, sequential insert performance was measured by running an application that used the following configuration:

- ▶ Three types of table spaces: partition-by-growth (PBG) table spaces, partition-by-range (PBR) table spaces with **PAGENUM RELATIVE (RPN)**, and PBR table spaces with **PAGENUM ABSOLUTE (APN)**.
- ▶ Member cluster, insert algorithm 1, and row-level locking (RLL).
- ▶ A JDBC application inserts into three tables with six non-unique indexes that are defined.
- ▶ A total of 10 partitions for the PBR table spaces.
- ▶ Multi-row (100) inserts and committing after each row set insert.
- ▶ A two-way data-sharing group with 100 concurrent threads in total.

The program first inserts data into the empty tables. Then, it deletes two-thirds of the rows to create “holes” in the pages. Finally, more rows are inserted into the populated tables to fill those existing holes.

As shown in Figure 4-7, the Db2 class 2 CPU time results demonstrate that Db2 13 can improve class 2 CPU time by up to 6.56%:

- ▶ For PBR table spaces with APN, the CPU time improved by 5.11% when inserting data into the populated tables.
- ▶ For PBR table spaces with relative page numbering (RPN), the class 2 CPU time was equivalent when inserting data into the populated tables.
- ▶ For PBG tables spaces, the CPU time improved by 6.56% when inserting data into the populated tables.

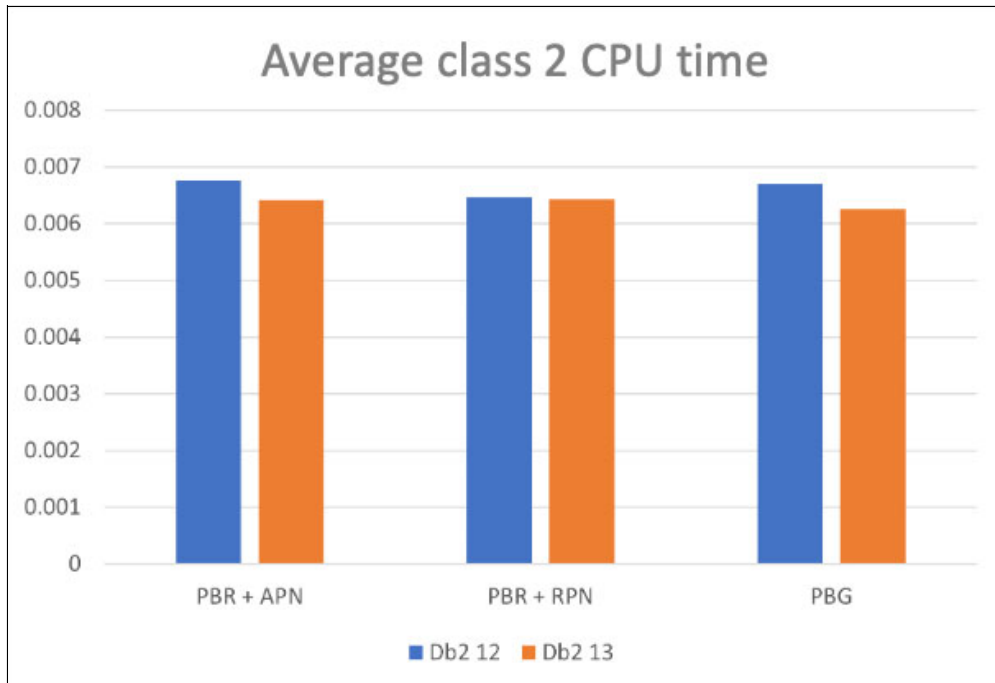


Figure 4-7 Archive sequential insert: Class 2 CPU time for PBR+APN, PBR+RPN, and PBG

All the CPU service time that is consumed on the sysplex from RMF workload activity reports, which is shown in Figure 4-8, demonstrates that Db2 13 can improve CPU time by up to 5.28%:

- ▶ For PBR table spaces with APN, the CPU service time improved by 3.99%.
- ▶ For PBR table spaces with RPN, the CPU service time improved by 5.28%.
- ▶ For PBG tables spaces, the CPU service time improved by 5.17%.

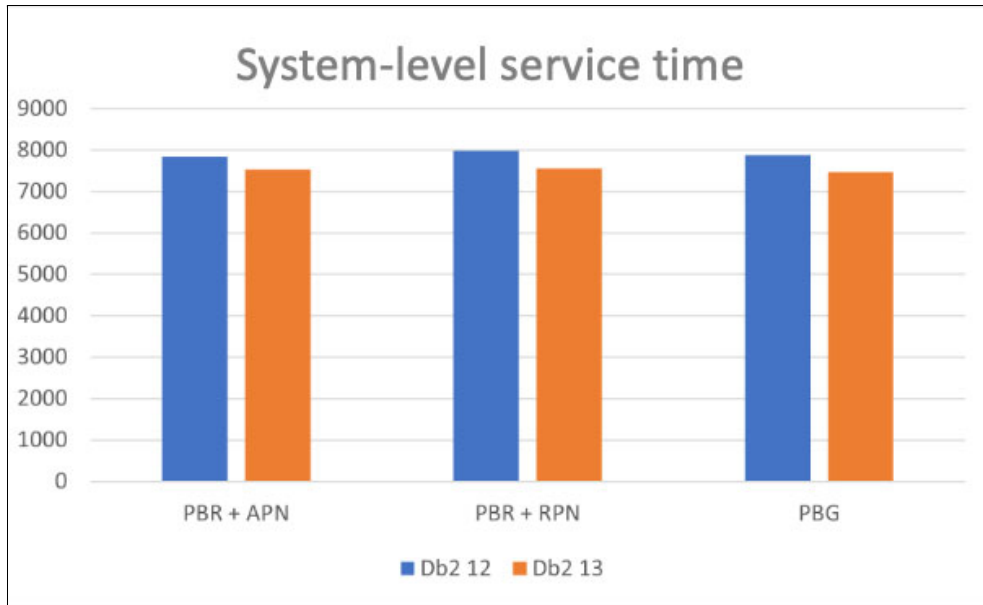


Figure 4-8 Archive sequential insert: System-level service time for PBR+APN, PBR+RPN, and PBG

The general improvement is from the index look-aside enhancement, which results in more than a 90% getpage reduction on index non-leaf pages for all the table space types that were tested. More improvement is seen in PBR RPN due to the reduction of false contention reductions.

The average class 2 elapsed time results, which are shown in Figure 4-9 on page 165, show less than a 1% improvement in Db2 13 compared with Db2 12.

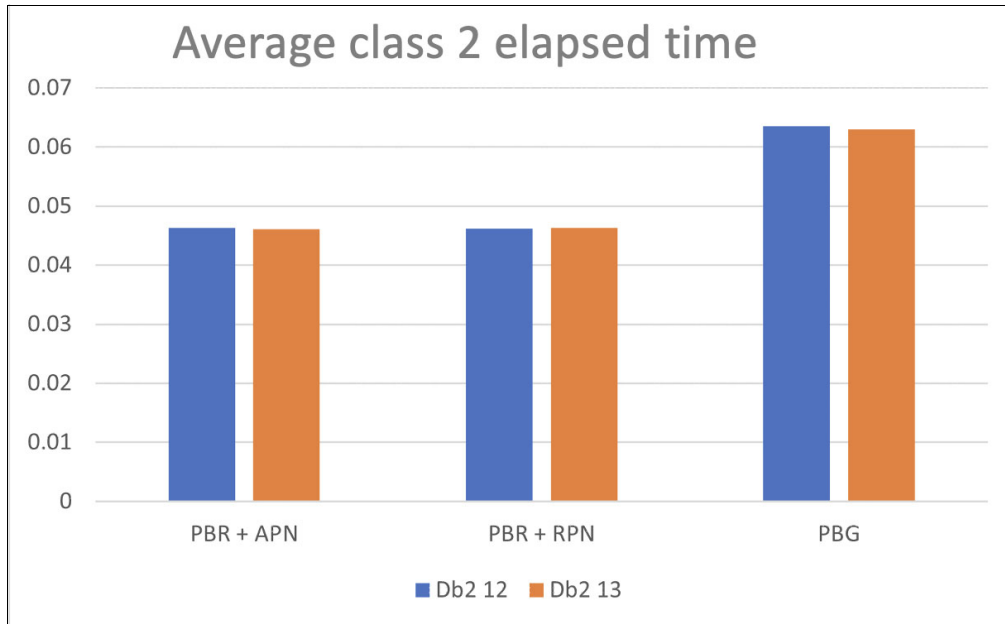


Figure 4-9 Archive sequential insert: Class 2 elapsed time for PBR+APN, PBR+RPN, and PBG

The throughput of rows that are inserted is divided by the CPU utilization to obtain the ITR. Figure 4-10 shows the ITR for the archive sequential insert scenario test run on Db2 13 and Db2 12 with the same user load and software and hardware configuration. It shows that Db2 13 performs 4.16%, 5.58%, and 5.46% better on PBR with APN, PBR with RPN, and PBG respectively.

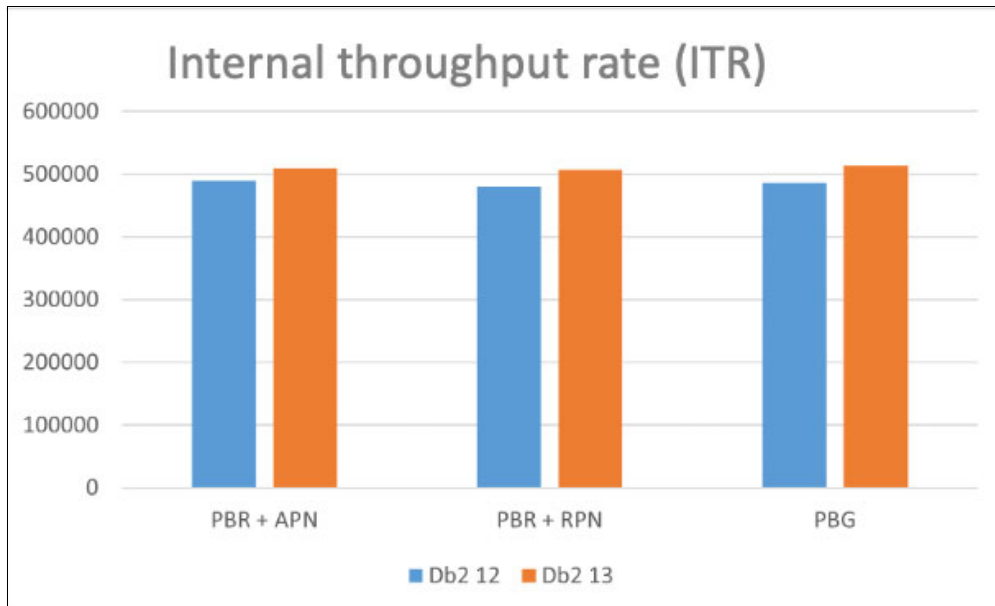


Figure 4-10 Archive sequential insert: Internal throughput rate for PBR+APN, PBR+RPN, and PBG

## Journal table insert

This scenario uses a sequential insert workload with the following characteristics:

- ▶ All the tests are done with multi-row insert (row set size 100 per commit) and RLL. A JDBC application inserts into three tables in a two-way data-sharing environment with 100 concurrent threads in total. Each table has 20 partitions (note the difference in the indexes between the journal inserts and archive sequential inserts, as described in “Archive sequential insert” on page 162).
- ▶ Test 1 was done with three PBR tables with **PAGENUM RELATIVE, MEMBER CLUSTER, APPEND NO, INSERT ALGORITHM 1 (IAG1)**, and five unique indexes and one non-unique index that were defined on three tables in total.
- ▶ Test 2 was done with three PBR tables with **PAGENUM RELATIVE, MEMBER CLUSTER, APPEND YES, INSERT ALGORITHM 1 (IAG1)**, and five unique indexes and one non-unique index that were defined on three tables in total.
- ▶ Test 3 was done with three PBR tables with **PAGENUM ABSOLUTE, MEMBER CLUSTER, APPEND NO, INSERT ALGORITHM 1 (IAG1)**, and no index.
- ▶ Test 4 was done with three PBR tables with **PAGENUM ABSOLUTE, MEMBER CLUSTER, APPEND NO, INSERT ALGORITHM 2 (IAG2)**, and one unique index that was defined on each table.
- ▶ Test 5 was done with three PBR tables with **PAGENUM RELATIVE, MEMBER CLUSTER, APPEND NO, INSERT ALGORITHM 1 (IAG1)**, and one unique index that was defined on each table.

The Db2 class 2 CPU time results, which are shown in Figure 4-11, show no significant CPU time changes for the journal table insert scenario in Db2 13 compared to Db2 12.

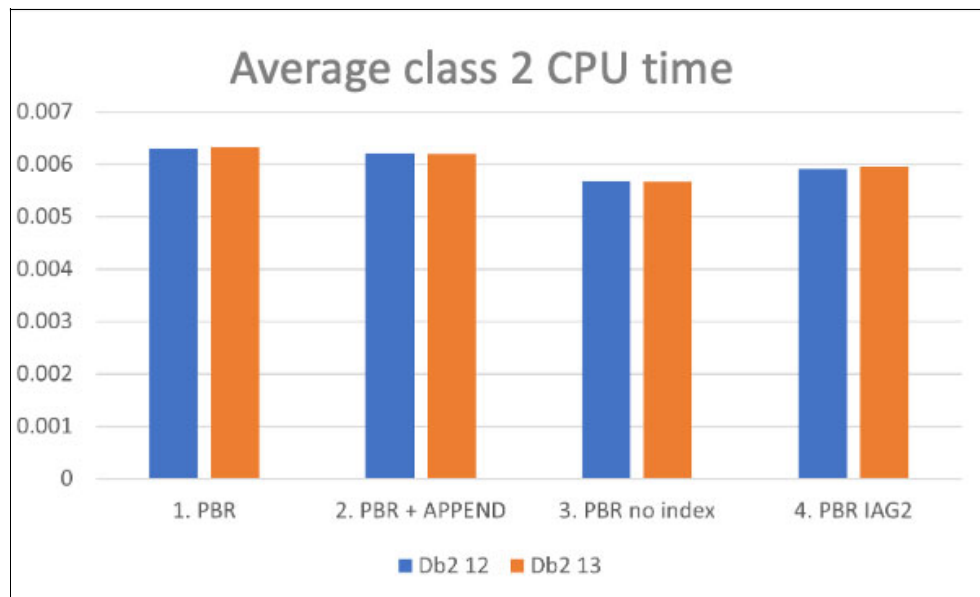


Figure 4-11 Journal table insert scenarios: Class 2 CPU time for Db2 12 versus Db2 13

All the CPU service time that was consumed on the sysplex from RMF workload activity reports, as shown in Figure 4-12 on page 167, demonstrate that Db2 13 can improve CPU time by 8.99% and 10.49% in workloads 1 and 2.



Figure 4-12 Journal table insert scenarios: System-level service time for Db2 12 versus Db2 13

A major factor that contributed to the result was an 84.9% reduction of false contention occurrences for workloads 1 and 2. Another factor was the index look-aside enhancement, which can result in up to an 88% getpage reduction for index non-leaf pages for workloads 1 and 2. In workloads 3 and 4, Db2 13 performed almost the same as Db2 12. Because fewer indexes were created, there was less opportunity for index look-aside and false contention savings.

The measurement results in Figure 4-13 show that an average class 2 elapsed time reduction of up to 5.1% was observed between Db2 12 and Db2 13.

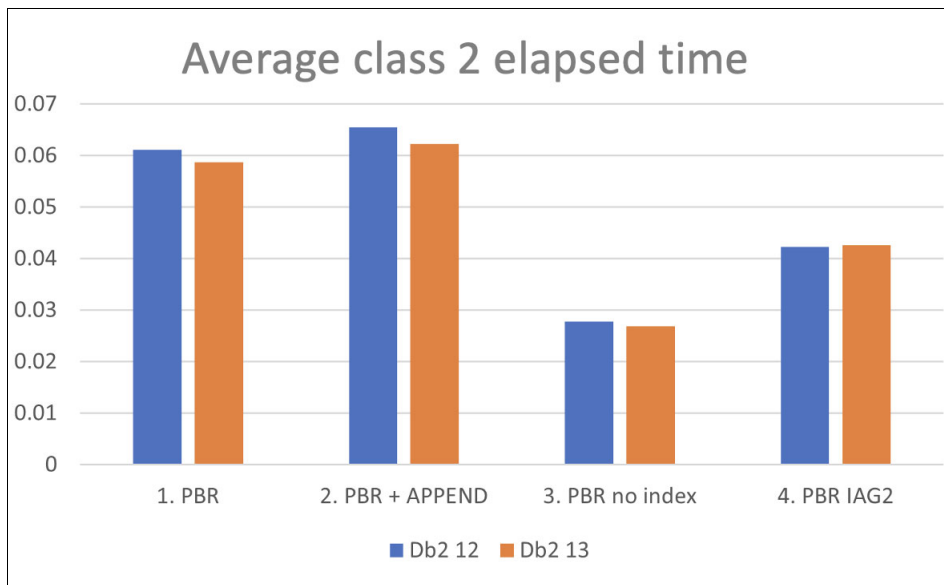


Figure 4-13 Journal table insert scenarios: Class 2 elapsed time for Db2 12 versus Db2 13

The throughput of rows that were inserted was divided by the CPU utilization to obtain the ITR. Figure 4-14 shows the ITR for the journal table insert scenario tests that ran on Db2 13 and Db2 12 with the same user load and software and hardware configuration. It shows that Db2 13 performed 9.88% and 11.71% better on workloads 1 and 2 respectively. In workloads 3 and 4, Db2 13 performed almost the same as Db2 12. Because there were fewer indexes that were created, there was less opportunity for index look-aside and false contention savings.

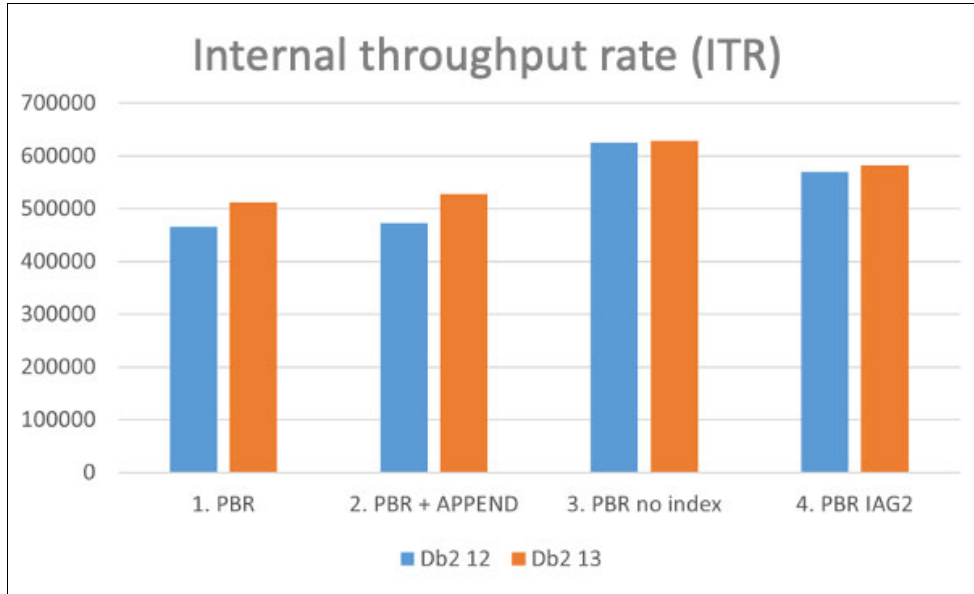


Figure 4-14 Journal table insert scenarios: Internal throughput rate for Db2 12 versus Db2 13

Db2 13 uses **IAG1** with the default setting for the **DEFAULT\_INSERT\_ALGORITHM** subsystem parameter. The results in Figure 4-15 and Figure 4-16 on page 169 show the average class 2 elapsed time and CPU time for workloads 4 and 5.

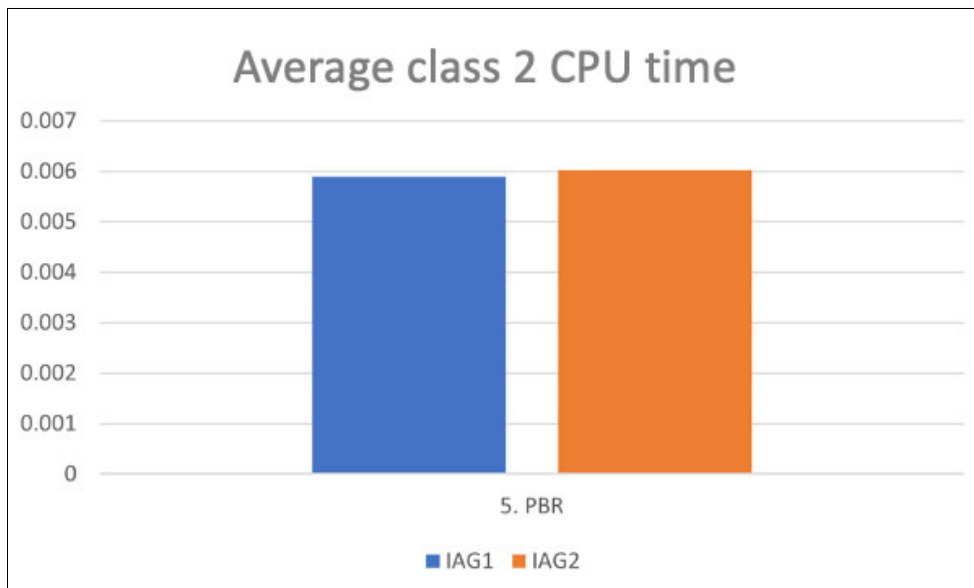


Figure 4-15 Db2 13 IAG1 versus IAG2: Class 2 CPU time



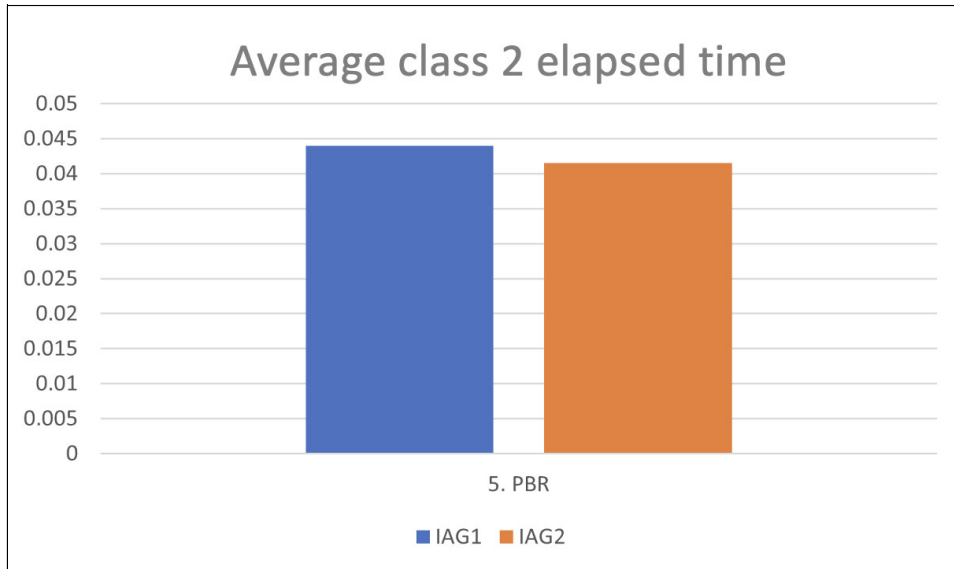


Figure 4-16 Db2 13 IAG1 versus IAG2: Class 2 elapsed time

These workloads insert data into three PBR tables with one index for each table. The average class 2 CPU time was 2.17% more when using **IAG2** than when using **IAG1** in Db2 13. The average class 2 elapsed time is 5.52% less when using **IAG2** than when using **IAG1** in Db2 13. Although using IAG2 helped to reduce data page latch suspension time, we saw Db2 latch suspension from index page split activities as the next bottleneck.

### Message queue insert and delete

This test scenario simulates a message queue management application. The Db2 table is treated as a queue. A task inserts some rows sequentially into the queue table while it also starts to process and delete rows at the front of the queue table. This scenario features the following configuration:

- ▶ A PBG table space
- ▶ RLL with and without the **MEMBER CLUSTER** option
- ▶ Multi-row inserts with 10 rows per commit
- ▶ Insert algorithm 1
- ▶ A two-way Db2 data-sharing group
- ▶ **APPEND NO**

The CPU time results in Figure 4-17 show a comparable class 2 CPU time for the two-way data-sharing benchmarks in Db2 13 compared to Db2 12.

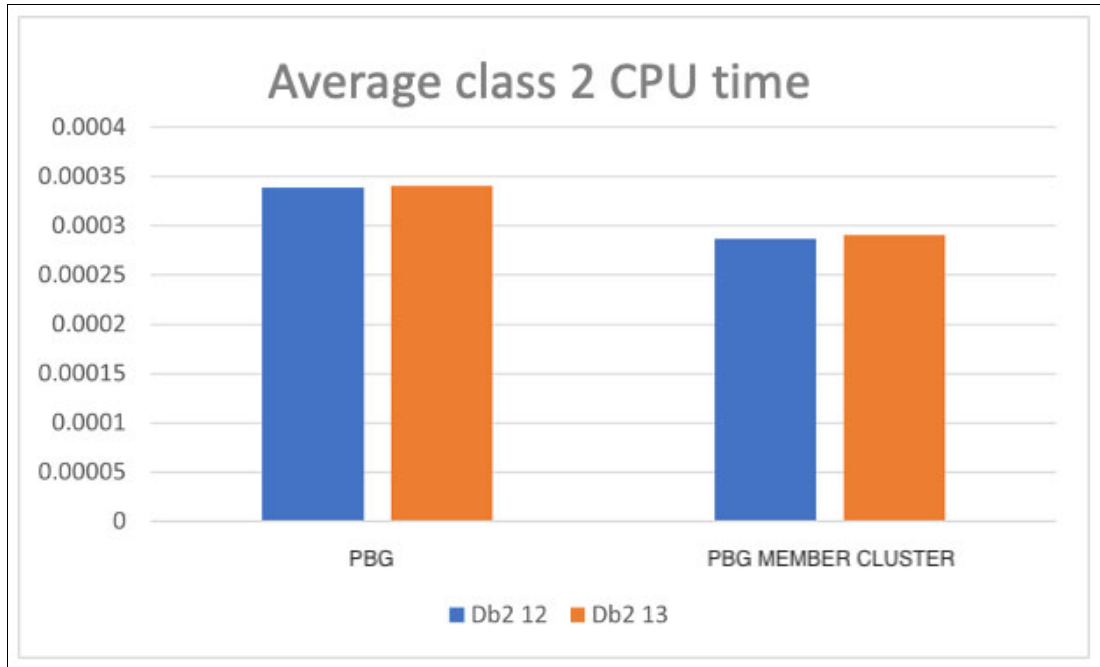


Figure 4-17 Message queue insert and delete scenarios: Class 2 CPU time for Db2 12 versus Db2 13

All the CPU service time that was consumed on the sysplex from RMF workload activity reports, as shown in Figure 4-18, show that the insert performance was almost the same in Db2 13 compared to Db2 12.

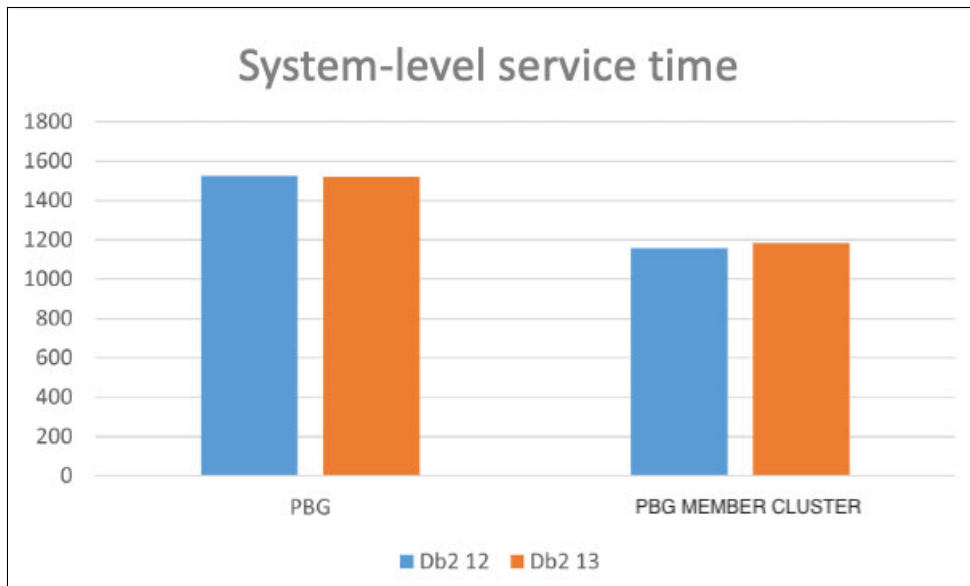


Figure 4-18 Message queue insert and delete scenarios: System-level service time

The results in Figure 4-19 on page 171 show that less than a 0.5% difference was observed in class 2 elapsed time between Db2 12 and Db2 13.

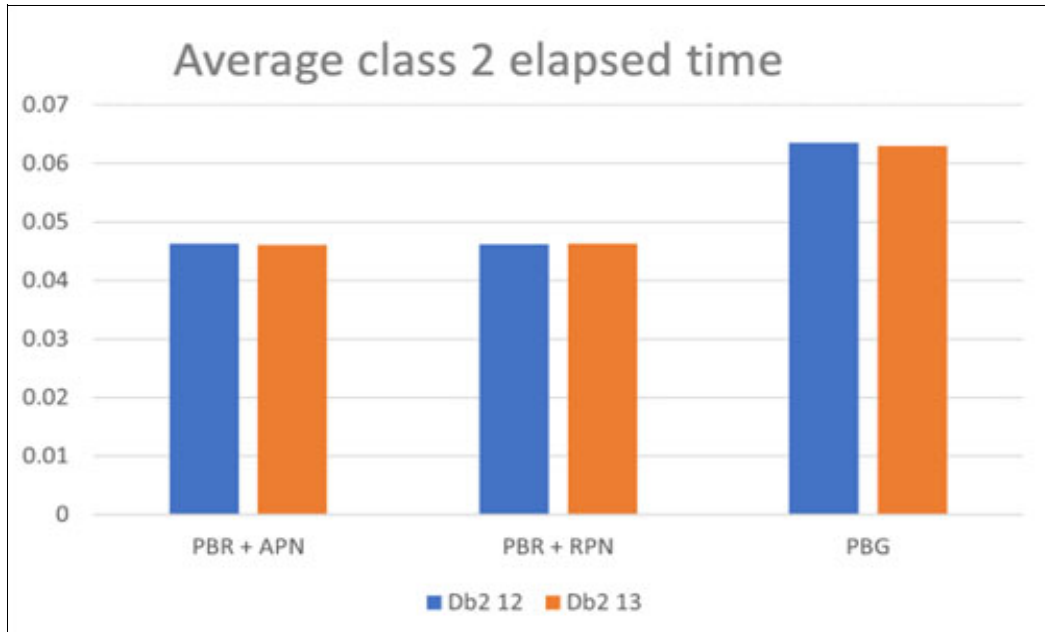


Figure 4-19 Message queue insert and delete scenarios: Class 2 elapsed time

The throughput of rows inserted was divided by the CPU utilization to obtain the ITR. Figure 4-20 shows the ITR for the message queue insert and delete scenario tests that ran on Db2 13 and Db2 12 with the same user load and software and hardware configuration. Again, it shows Db2 13 performs almost the same as Db2 12.

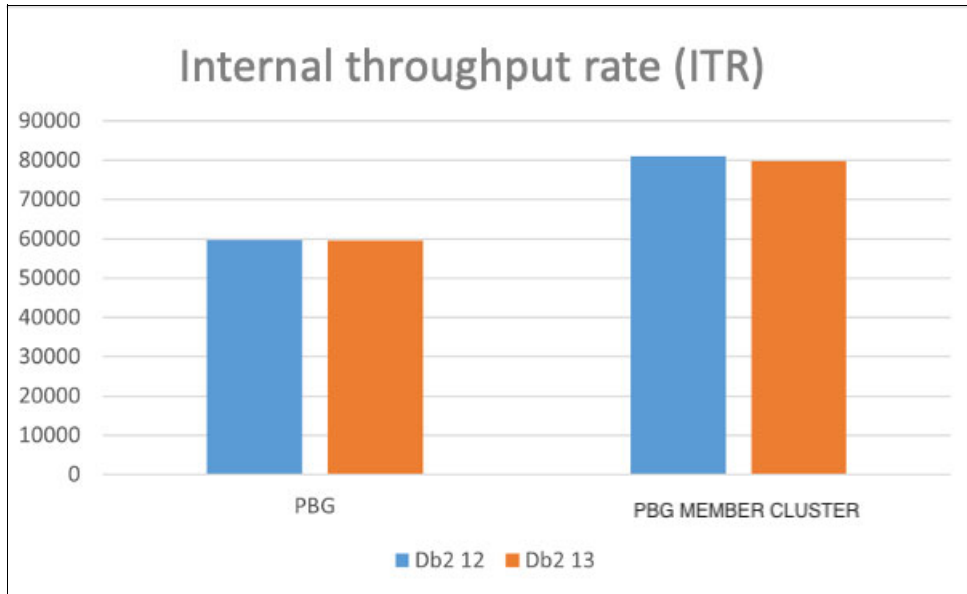


Figure 4-20 Message queue insert and delete scenarios: Internal throughput rate

## Random insert and delete

This workload scenario uses an application with a random insert and delete pattern. The benchmark setup uses the following configuration:

- ▶ Two types of tables spaces:
  - PBG without the **MEMBER CLUSTER** option
  - PBR with **PAGENUM RELATIVE** without the **MEMBER CLUSTER** option
- ▶ Page-level locking.
- ▶ A JDBC application inserting data into one table with one unique clustering index with 100 partitions for the PBR case.
- ▶ Insert algorithm 1.
- ▶ **APPEND NO.**
- ▶ A two-way data-sharing group that concurrently runs 180 threads in total, evenly spread between both members, which are performing inserts, and 120 threads running deletes in total, evenly spread between both members.

During the test runs, the table is first seeded with 40 million rows. Then, 4 million rows are randomly inserted. Concurrently, 2.7 million rows are randomly deleted. The inserts are done concurrently by 180 threads, and the deletes are done concurrently by 120 threads that are spread across the two Db2 data-sharing members.

The Db2 class 2 CPU time results, as shown in Figure 4-21, show the following results:

- ▶ For PBR table spaces, a 4.9% class 2 CPU time saving was observed between the Db2 13 and Db2 12 test runs.
- ▶ For PBG table spaces, an 82% class 2 CPU time saving was observed between the Db2 13 and Db2 12 test runs. A 28% getpage reduction was observed due to the index look-aside enhancement on index non-leaf pages. The enhanced insert candidate page searching algorithm is another contributor. For more information about this enhancement, see 7.1.2, “Retrying a search of previously failed partitions” on page 242.

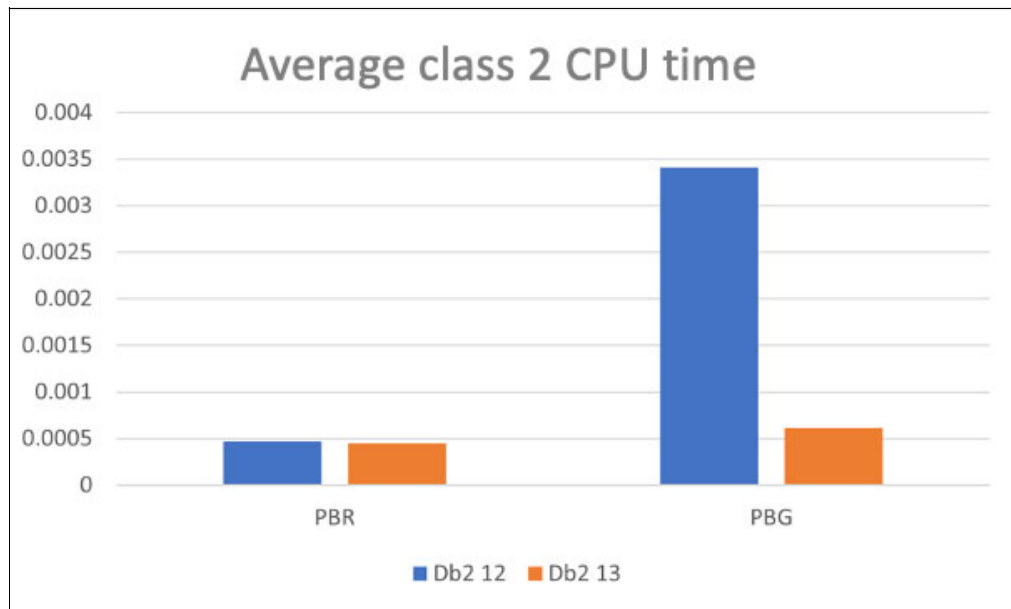


Figure 4-21 Random insert and delete scenarios: Class 2 CPU time for PBR and PBG

All the CPU service time that was consumed on the sysplex from RMF workload activity reports, as shown in Figure 4-22, shows that the overall CPU usage was 5.3% and 20.01% less on PBR and PBG respectively in Db2 13 compared to Db2 12.

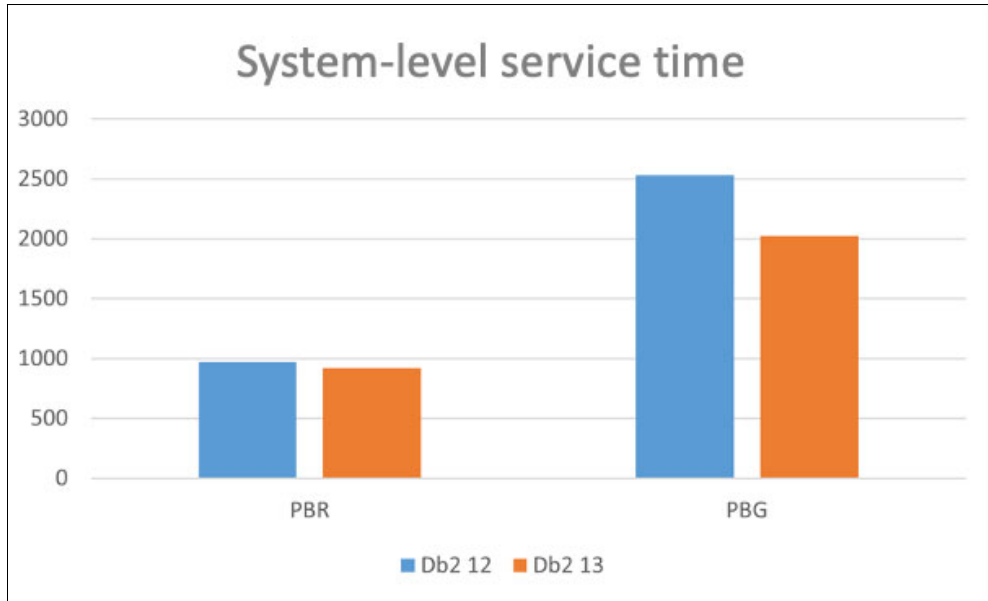


Figure 4-22 Random insert and delete scenarios: System-level service time for PBR and PBG

The Db2 class 2 elapsed time results, which are shown in Figure 4-23, show the following results:

- ▶ For PBR table spaces, no notable change was observed in Db2 13 compared to Db2 12. These transactions are not CPU bound so no significant increase in throughput rate was expected.
- ▶ For PBG table spaces, a 75% class 2 elapsed time saving was observed between the Db2 13 and Db2 12 test runs.

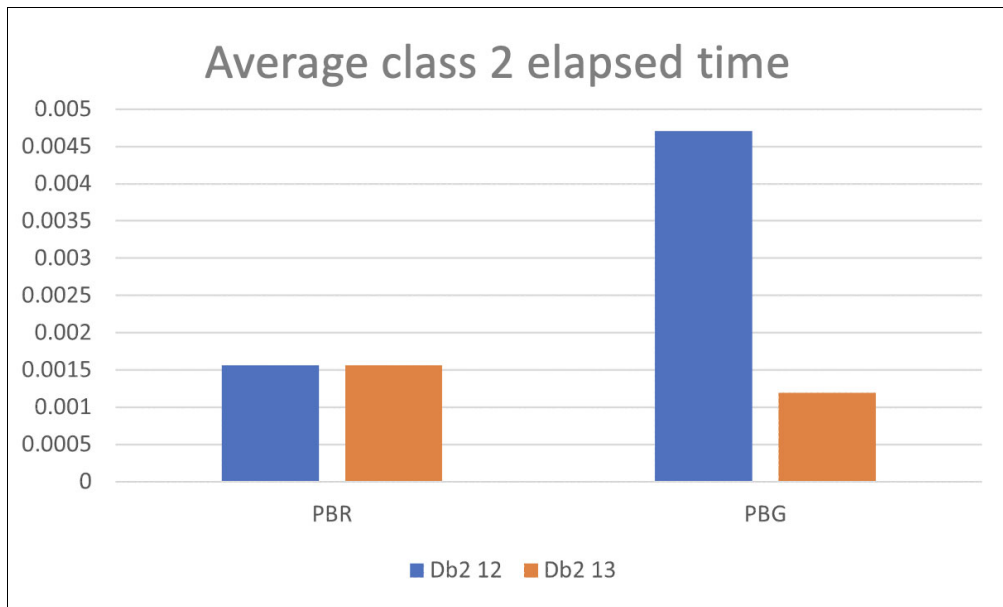


Figure 4-23 Random insert and delete scenarios: Class 2 elapsed time for PBR and PBG

The throughput of rows inserted was divided by the CPU utilization to obtain the ITR. Figure 4-24 shows the ITR for random insert and delete scenario tests that ran on Db2 13 and Db2 12 with the same user load and software and hardware configuration. Again, it shows Db2 13 performed 5.59% and 25.02% better on PBR and PBG compared with Db2 12.

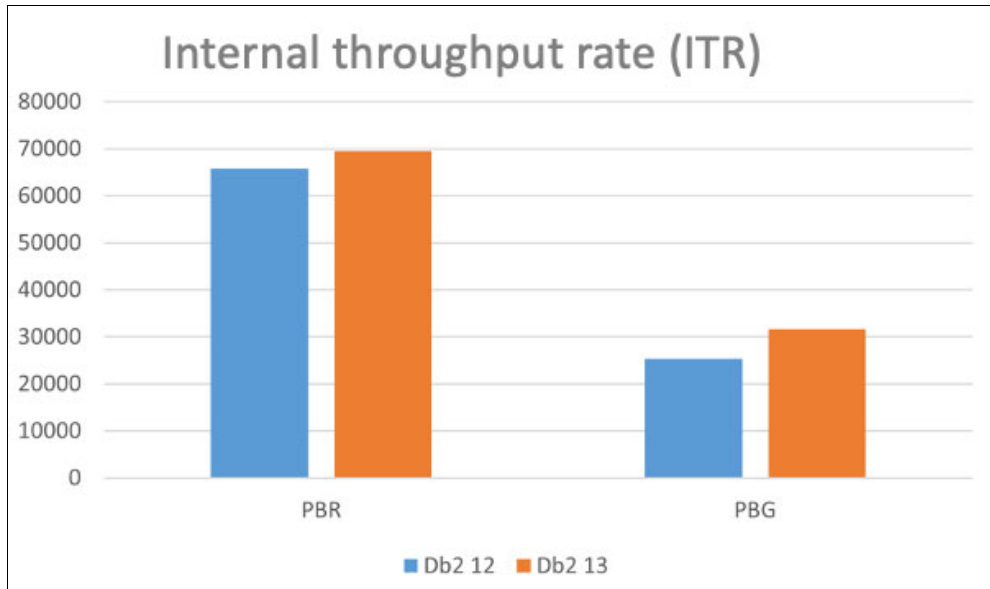


Figure 4-24 Random insert and delete scenarios: Internal throughput rate for PBR and PBG

### Massive simple row insert

This workload scenario uses an SQLJ application with a sequential insert pattern. The benchmark setup uses the following configuration:

- ▶ PBR table spaces with **MEMBER CLUSTER** and **PAGENUM ABSOLUTE**.
- ▶ Page-level locking.
- ▶ One table without index.
- ▶ 200 partitions.
- ▶ Insert algorithm 1.
- ▶ **APPEND NO**.
- ▶ A two-way data-sharing group that runs 200 concurrent threads, which are evenly distributed between the members, with each thread inserting 100 rows per commit.

The CPU time results, as shown in Figure 4-25 on page 175, show a comparable class 2 CPU time for the two-way data-sharing benchmarks in Db2 13 compared to Db2 12.

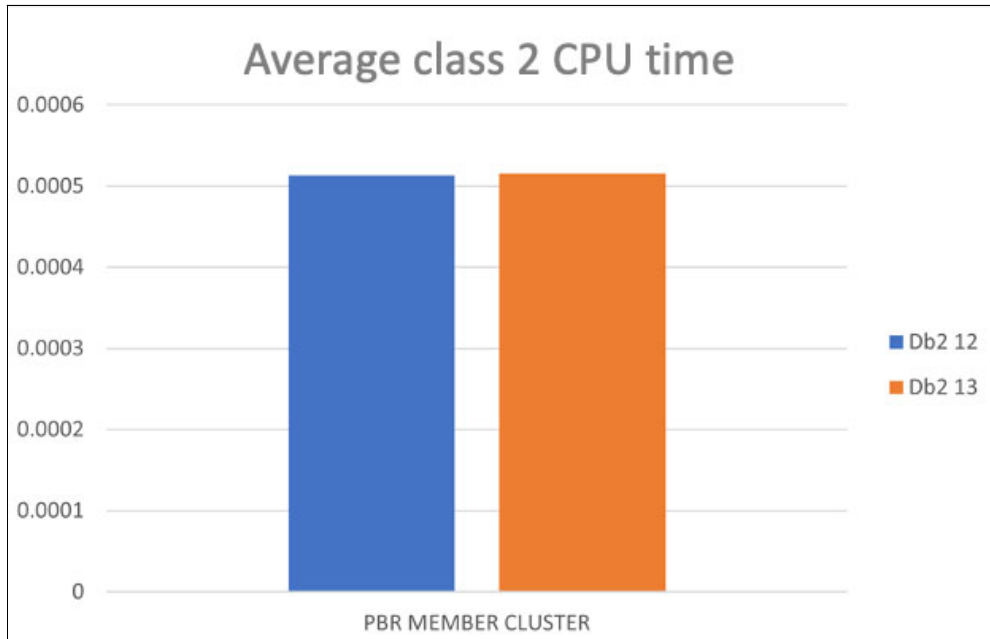


Figure 4-25 Massive simple row insert: Class 2 CPU for Db2 13 versus Db2 12

All the CPU service time that was consumed on the sysplex from RMF workload activity reports, as shown in Figure 4-26, show that the overall CPU usage is 3.54% in Db2 13 compared to Db2 12.



Figure 4-26 Massive simple row insert: System-level service time for Db2 13 versus Db2 12

This massive simple row insert scenario is not a realistic customer workload because it does not use any indexes. It is used to evaluate any differences in the insert throughput rate between Db2 12 and 13. The results are shown in Figure 4-27.

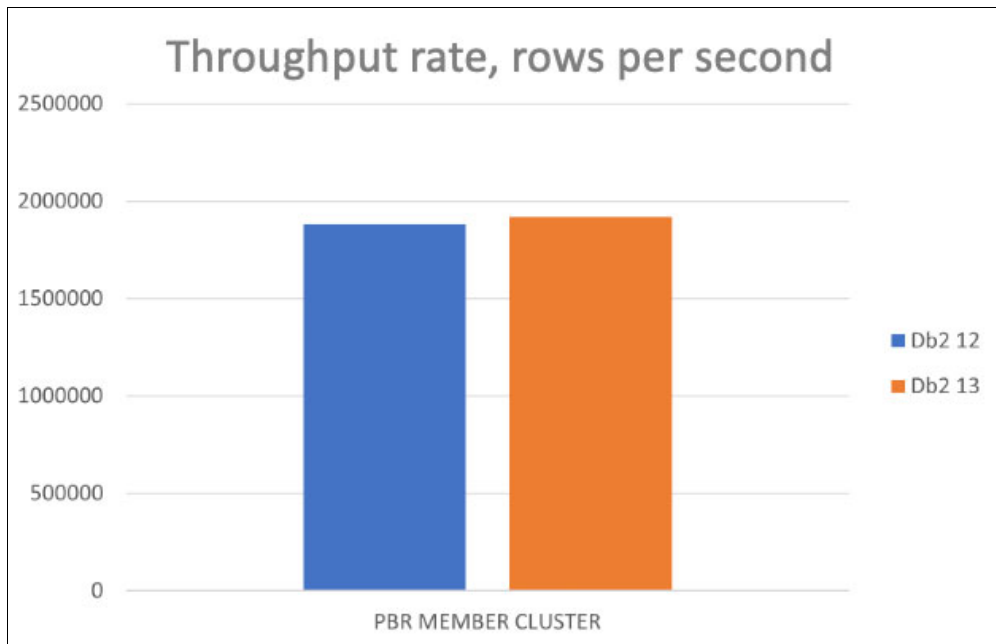


Figure 4-27 Massive simple row insert: Throughput rate for Db2 13 versus Db2 12

Because this application evenly spreads threads that insert data into different partitions and because no index is created on the table, there is less opportunity for savings that are related to false contentions the index look-aside enhancement, so the CPU usage is close between Db2 12 and Db2 13. Meanwhile, the external throughput rate from the statistic report in Db2 13 is 1.95% greater than for Db2 12.

The throughput of rows that were inserted was divided by the CPU utilization to obtain the ITR. Figure 4-28 on page 177 shows the ITR for the massive simple row insert scenario test runs on Db2 13 and Db2 12 with the same user load and software and hardware configuration. It shows that Db2 13 performs almost the same when compared with Db2 12.



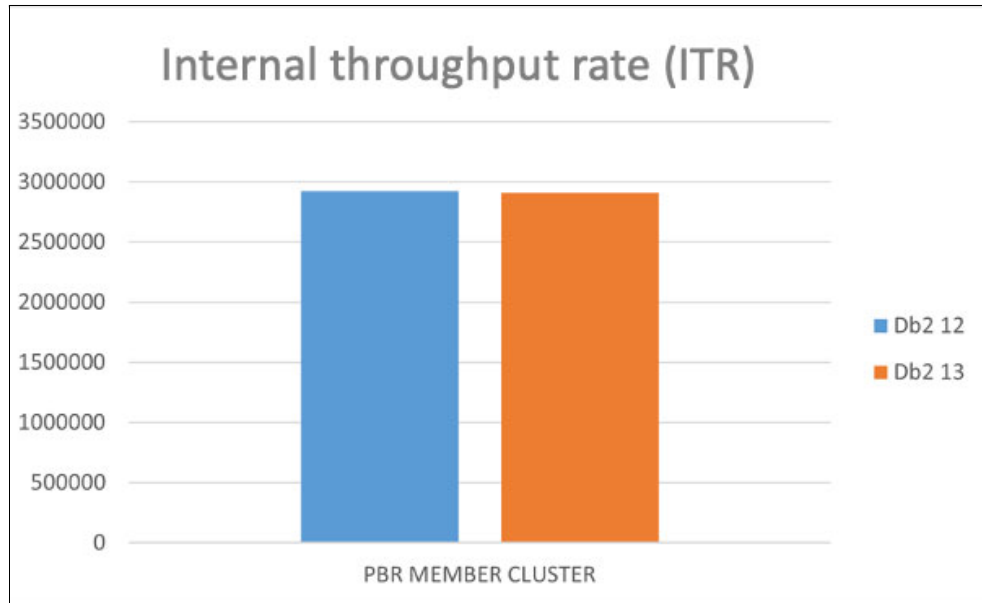


Figure 4-28 Massive simple row insert: Internal throughput rate for Db2 13 versus Db2 12

## 4.6 Relational transaction workload (CICS and Db2)

The relational transaction workload (RTW) is a standard workload that is used by the CICS performance team to assess changes in performance characteristics for new CICS releases for applications accessing Db2. The Db2 team employed the RTW workload as one of its performance test benchmark workloads during the development of Db2 12 to get better performance analysis coverage for CICS-Db2 type workloads. The workload was modified slightly compared to the version that the CICS team uses to better reflect the Db2 behavior of the workload.

The RTW workload has the following characteristics:

- ▶ All programs that make up the workload are written in COBOL.
- ▶ The workload uses 20 tables, 11 non-unique indexes, and 16 unique indexes.
- ▶ It issues on average 200 Db2 calls per transaction.
- ▶ The transaction mix consists of 26% select, 3% insert, 7% update, 3% delete, 6% open cursor, 36% fetch cursor, and 6% close cursor operations.

### 4.6.1 Performance measurement

As a part of Db2 13 performance evaluation process, the non-data-sharing and two-way data-sharing version of the RTW workload were run, and the results were compared to the Db2 12 performance numbers. The measurements used Db2 12 FL 510 as the baseline. Then, the workload performance for Db2 13 at FL 500 was measured. The main new feature that affects the performance of the RTW workload is the expanded FTB support for non-unique indexes.

## Measurement environment

The following environment configuration was used for the non-data-sharing RTW workload:

- ▶ One IBM z15 LPAR with eight general CPs and two zIIPs
- ▶ Four CICS regions running CICS Transaction Server (CICS TS) 5.6
- ▶ A non-data-sharing Db2 subsystem
- ▶ z/OS 2.5
- ▶ DS8870 direct access storage device (DASD) controller
- ▶ Teleprocessing Network Simulator (TPNS) running on another IBM z15 LPAR with 4 general CPs to drive the workload

The environment that was used by the two-way data-sharing RTW workload was as follows:

- ▶ Two IBM z15 LPARs with eight general CPs and two zIIPs each
- ▶ Eight CICS regions running CICS TS 5.6 with four regions running on each of the LPARs
- ▶ z/OS 2.5
- ▶ DS8870 DASD controller
- ▶ TPNS running on another IBM z15 LPAR with four general CPs to drive the workload
- ▶ Two IBM z15 Internal Coupling Facility (ICF) LPARs running CFLEVEL 25 CFCC with each connected to the z/OS LPARs through two CS5 channels

## Results

This section describes the measurement results of the RTW workload for both non-data sharing and two-way data sharing.

### ***RTW results: Non-data sharing***

From the performance measurement numbers, the following general observations were seen while comparing the non-data-sharing RTW workload performance runs for Db2 13 and Db2 12:

- ▶ Class 2 CPU usage was reduced by approximately 3% per commit, which was attributed mainly to the getpage reduction of 40% because of the non-unique index support by fast index traversal (FTB).
- ▶ Address space CPU per commit was reduced by approximately 2%. The main source of this reduction came from *ssnm*MSTR System Recovery Boost (SRB) CPU time. The CPU savings were the result of optimized above-the-bar (ATB) storage management in Db2 13 where it issued fewer **IAVR64 DISCARDATA** requests compared to Db2 12 when `REALSTORGE_MANAGEMENT=AUTO` was used, which was the option that is used for the Db2 12 measurement.
- ▶ With the same transaction rates, the average CP CPU BUSY% during the measurement periods dropped by approximately 2%. zIIP CPU had a slight increase that can be ignored.
- ▶ The real storage numbers that were obtained from the statistics reports indicate that the total real storage usage for the Db2 13 test, even with the expanded FTB usage, was only 171 MB (approximately 1% and well within measurement noise level) more than the Db2 12 test for the RTW non-data-sharing workload.

**Note:** During normal operation (the LPAR is not running low on real storage), the ATB real storage Db2 releases to z/OS are still charged to Db2 until they are reused by other address spaces. This situation makes the 64 BIT REAL STORAGE IN USE numbers in Db2 records inaccurate, and often reflect the high-water-mark storage usage. This situation is applicable to both Db2 12 and Db2 13.

Figure 4-29 shows more details about the RTW non-data-sharing measurements.

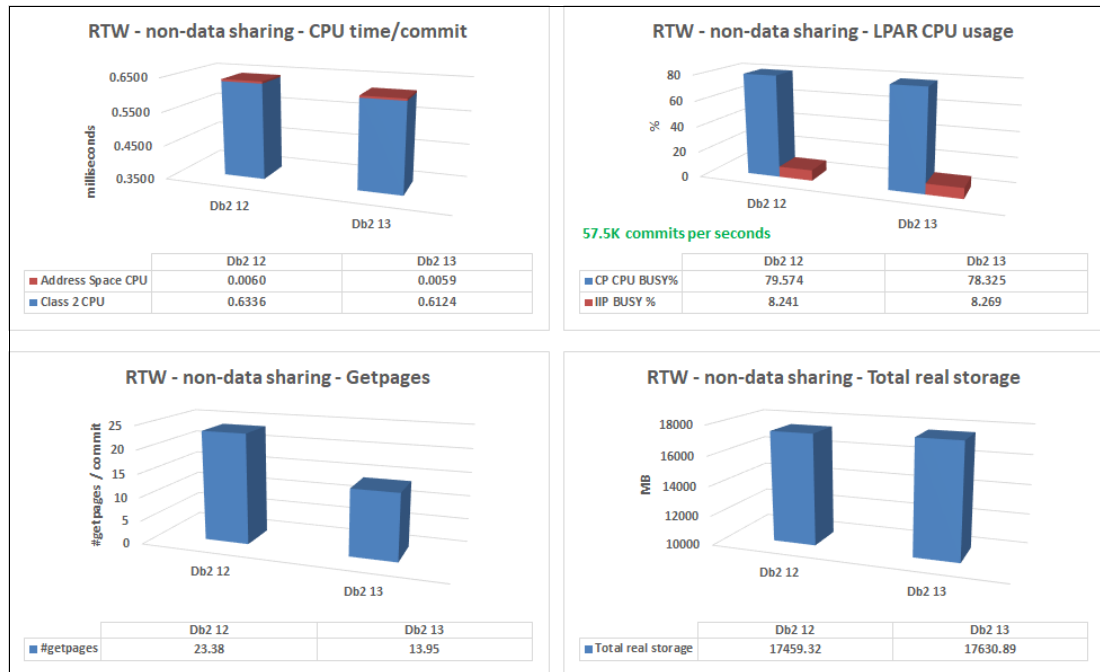


Figure 4-29 RTW: Non-data-sharing measurement results for Db2 13 versus Db2 12

### RTW results: Two-way data sharing

For the two-way data-sharing RTW workloads, we made the following general observations when comparing the performance numbers of the Db2 13 and Db2 12 tests (all numbers are calculated and presented as a group average):

- ▶ Class 2 CPU usage was reduced by approximately 2% per commit attributing mainly to the getpage reduction of 40% because of non-unique indexes being supported by fast index traversal (FTB). The class 2 CPU reduction is less than the non-data-sharing tests because the getpage activity accounts for a smaller portion of the class 2 CPU time in a data-sharing environment.
- ▶ Address space CPU per commit was reduced by approximately 2%. The main source of the reduction came from the *ssnm*MSTR SRB CPU time. The CPU saving was the result of optimized ATB storage management in Db2 13. The Db2 12 measurement was conducted by using the default REALSTORAGE\_MANAGENT = AUTO setting.
- ▶ With the same transaction rates, the average CP CPU BUSY% during the measurement periods dropped by approximately 1%. zIIP CPU had some slight increase that can be ignored.
- ▶ The real storage numbers, which were obtained from the statistics reports, indicate that the total real storage usage on average for a Db2 member for the Db2 13 test is 44 MB more than in the Db2 12 test.

**Note:** During normal operation, meaning that the LPAR is not running low on real storage, the ATB real storage Db2 that is released to z/OS is still charged to Db2 until it is reused by other address spaces. This situation makes the 64 BIT REAL STORAGE IN USE numbers Db2 records inaccurate, and often reflect the high-water-mark storage usage. This situation is applicable to both Db2 12 and Db2 13.

Figure 4-30 shows more detail about the two-way data-sharing measurements of the RTW.

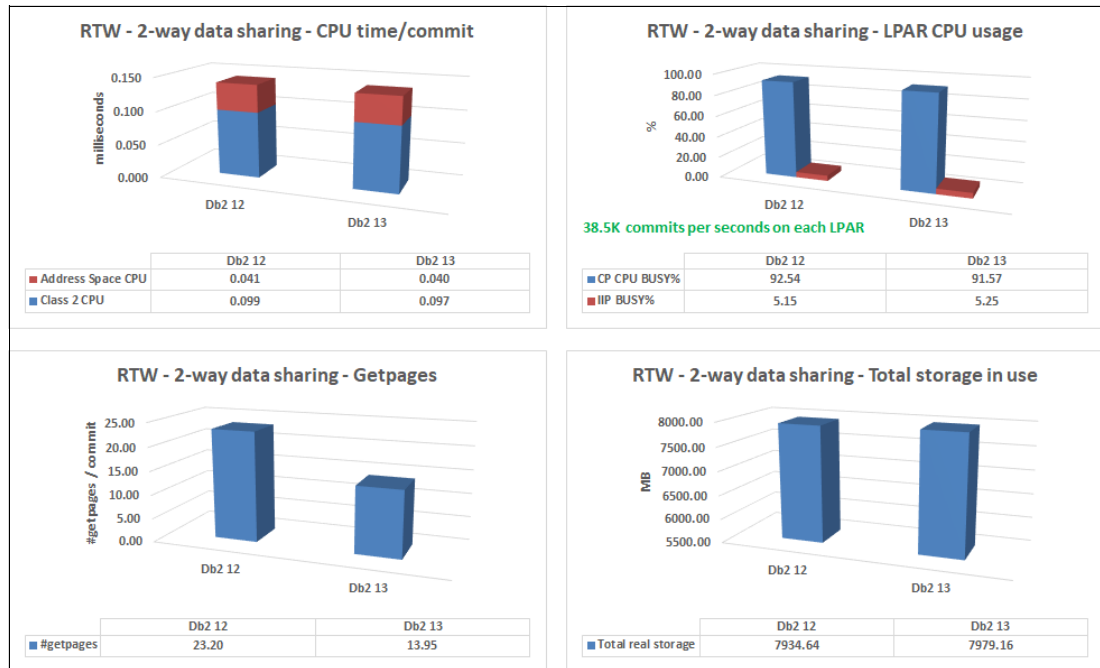


Figure 4-30 RTW: Two-way data-sharing measurement results comparing Db2 13 to Db2 12

### Summary of results

After migrating from Db2 12 to Db2 13, some performance improvement still occurs for OLTP type of workloads with more indexes that are supported by fast index traversal (FTB). Also, with the storage management optimization of ATB storage, the *ssnmMSTR* address space SRB CPU time might have some CPU time reduction.

## 4.7 IBM query regression workload

The IBM query regression workloads are a mix of complex OLTP, real-time analytics, and analytical data warehouse queries with industry benchmark data and masked data from customers. For more information about the workload, see Appendix A, “IBM Db2 workloads” on page 403.

A total of nine Db2 internal query regression workloads were measured to compare Db2 13 to Db2 12. Those workloads were combined with several sets of queries for complex transactional processing and real-time analytics in Db2.

## 4.7.1 Characteristics of query workloads and testing environment

The complex query workloads have the following characteristics:

- ▶ Customer workload 1: Complex queries with **JOIN** and **GROUP BY** table functions.
- ▶ Customer workload 2: Join, table functions, sorts, and IN list.
- ▶ Customer workload 3: Complex real-time analytics (outer joins, table expression, CASE, and DGTs).
- ▶ TPCD workload: Complex, long running queries against large complex data structures.
- ▶ Modified TPC-H workload: Suite of business-oriented, ad hoc dynamic queries.
- ▶ Modified TPC-H SP workload: Suite of business-oriented, ad hoc static queries that run in Db2 stored procedures.
- ▶ SAP CDS FIN workload: Selective, real-time analytic queries that include large **GROUP BY** sorts and workfile usage, and sparse index on **VARGRAPHIC**.
- ▶ SAP CDS FIORI workload: Selective, complex, and real-time analytics queries that include outer joins, **UNION ALL**, and user-defined functions (UDFs) and table functions.
- ▶ IBM BIDAY workload: Complex SQL.

The IBM query regression workload was measured by using a non-data-sharing configuration with Db2 13 at FL 501. Db2 12 at FL 505 measurements were used as the baseline.

All performance measurements were performed in the following environment:

- ▶ IBM z15 LPAR with four general CPs and three zIIPs
- ▶ z/OS 2.4

## 4.7.2 Performance results

Figure 4-31 shows the query workloads or set of queries that demonstrate a 0.22 - 2.31% CPU time reduction compared to Db2 12, which is at noise level.

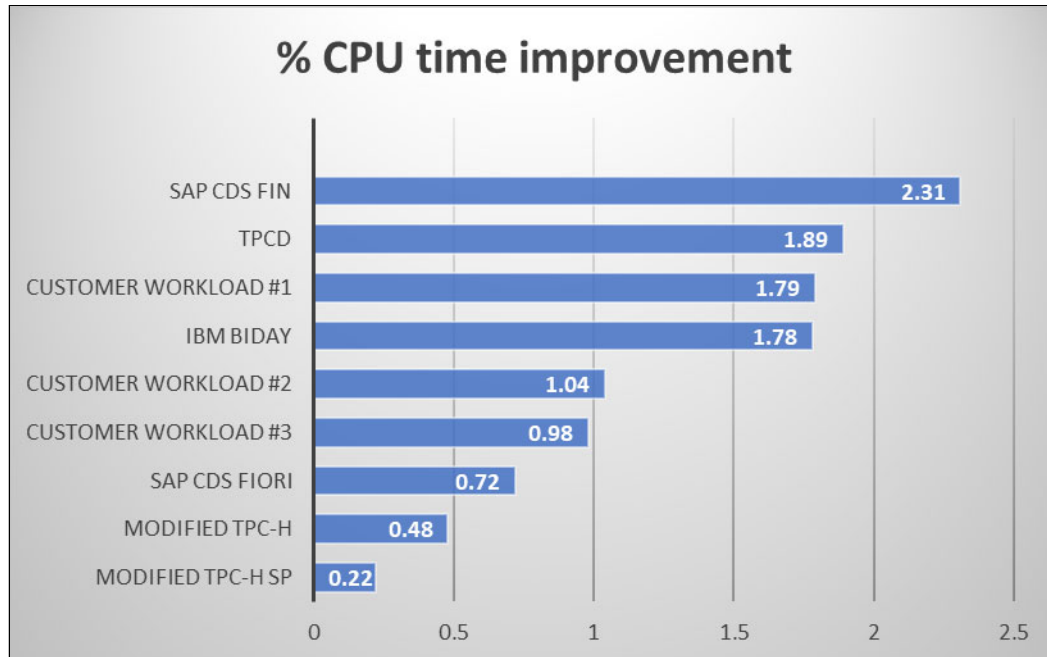


Figure 4-31 IBM complex query workload: CPU time improvements Db2 13 versus Db2 12

In Figure 4-31, the Db2 class 2 CPU time for the IBM query workloads in Db2 13 almost is the same as with Db2 12.

In this performance measurement, there are no access path changes in Db2 13 compared with Db2 12.

Many Db2 users tend to use **APREUSE** for their static SQL statements to keep their known stable access paths during migration to a new Db2 release. Even though Db2 13 introduces no access path enhancements, this approach is still a best practice to reduce the risk from the migration.

In Db2 13, more indexes are eligible for in-memory fast index traversal due to the default parameter setting of **FTB\_NON\_UNIQUE\_INDEX** changing from N0 to YES and changing the index key size limitations. Although the IBM complex query workloads that are described here did not leverage fast index traversal, it is possible to see a reduction in buffer pool getpages in other (customer) query workloads because more indexes now eligible for in-memory fast traversal. These set of query workloads did not leverage FTB because the objects are restarted at every query to evaluate an accurate impact from any access path updates.

## 4.8 SAP banking workloads

Using SAP with Db2 is a typical use case, especially in the financial and banking space. This section compares Db2 13 performance to Db2 12 by using two SAP banking workloads: SAP Day Posting and SAP Account Settlements. These two workloads are part of the SAP core banking application suite.

SAP Day Posting workload is an interactive OLTP workload. The workload simulates users interactively running 15 dialog steps. The key metrics are the ITR and response time. The workload has intensive insert, update, delete, and simple SQL statements. The data access pattern is random.

SAP Account Settlements workload is a batch workload that balances all the accounts in the database. The key metrics are elapsed time and ITR. The workload has intensive insert, update, delete, and SQL statements that are simple to complex. The data access pattern is mostly sequential.

SAP environments typically have the following three tiers:

**Presentation Server** Where the users submit requests. These requests are sent to the SAP application server. In this test environment, the users are simulated on the same machine as the presentation server.

**Application Server** Where the SAP application runs. This SAP server receives requests from the presentation server and runs them. It sends requests to the database server to obtain the necessary data.

**Database Server** In our tests, the Db2 z/OS database server, which handles database requests coming from the SAP application server.

The SAP banking database that is used for these tests consists of 100 million accounts, which is in scale with the largest banks in the world.

### 4.8.1 Scalability test environment

In this study, we compared the scalability of Db2 13 to Db2 12 by using the SAP Day Posting workload. We used the following setup for the three SAP tiers:

- ▶ Presentation Server: IBM AIX® 7.2. IBM POWER7 processor with 32 cores.
- ▶ Application Server: Linux on IBM zSystems running Red Hat 8.2, nine IBM z15 LPARs, and 66 Integrated Facility on Linux (IFL) cores. The application server and database server are on same IBM z15.
- ▶ Database Server: z/OS 2.4 with one IBM z15 LPAR. The number of Central Processors (CPs) varies between 2, 6 and 12 during the scalability tests.

### 4.8.2 Scalability performance results

To measure the scalability of Db2, the SAP Day Posting workload runs with three different CP configurations, that is, 2 cores, 6 cores, and 12 cores. On the z/OS Database Server LPAR, the throughput is divided by the CPU utilization to obtain the ITR. The user load is increased as the number of cores increase, so the CPU utilization is around 75 - 85% in each scenario.

Figure 4-32 shows that as we vary the number of cores and increase the user load, the scalability of Db2 13 at FL 500 is like Db2 12 at FL 510. The system is tuned so that the response time is close to 1 second in all cases.

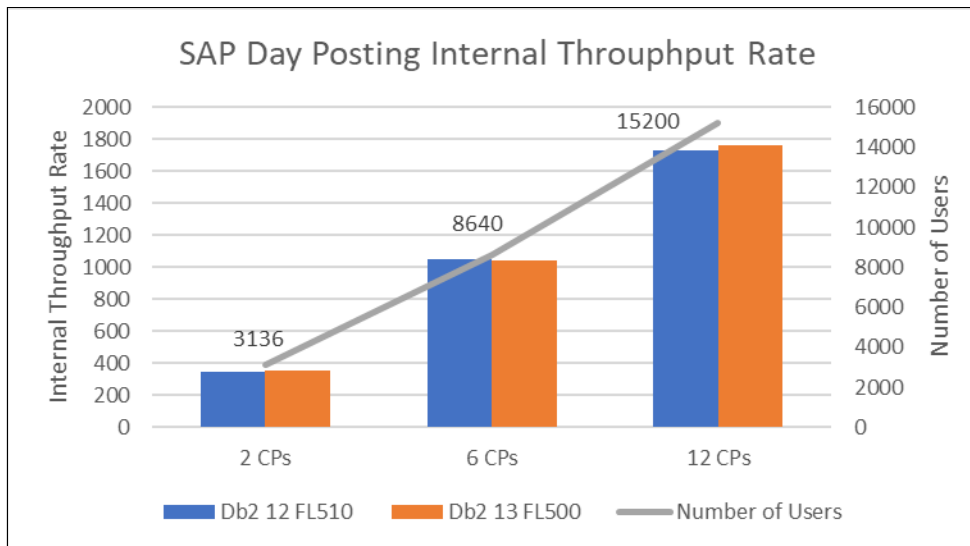


Figure 4-32 SAP Day Posting scalability internal throughput rate

The CPU utilization for each of the core configurations is shown in Figure 4-33. It shows that for a core configuration, the CPU utilization difference between Db2 13 and Db2 12 is 0 - 2%.

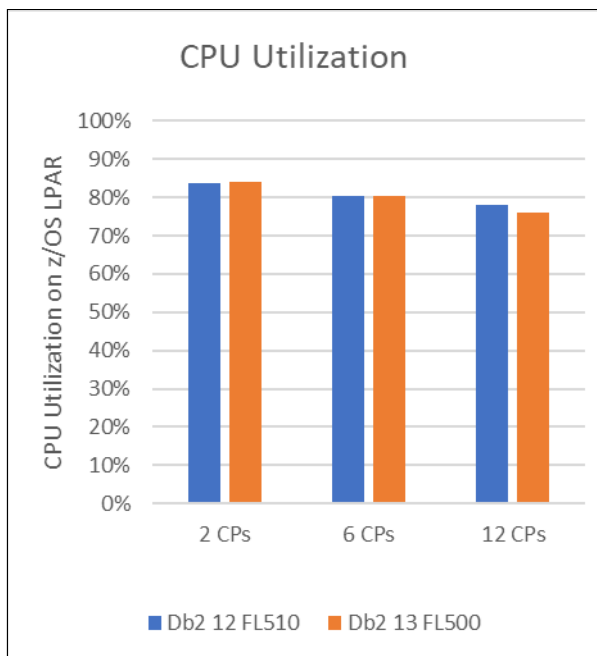


Figure 4-33 SAP Day Posting scalability test CPU utilization



### 4.8.3 Data-sharing test environment

The SAP Day Posting and SAP Account Settlement workloads also were compared between Db2 13 and Db2 12 by using a two-member data-sharing environment. The following setup was used for the three SAP tiers:

- ▶ Presentation server: AIX 7.2. IBM POWER7 with 32 cores for the presentation server.
- ▶ Application server: Linux on IBM zSystems running Red Hat 8.2, nine z15 LPARs, and 66 IFL cores. The application server and database server are on the same IBM z15.
- ▶ Database server: Two IBM z15 LPARs are used with four CPs for each of the two z/OS LPARs, with one Db2 data-sharing member on each z/OS LPAR. Two CF LPARs with 6 ICF cores each are used, and asynchronously lock duplexing is enabled. z/OS 2.4 is used for these tests.

The SAP Day Posting and SAP Account Settlement workloads run with Db2 13 FL 500 and Db2 12 FL 510 by using this configuration.

### 4.8.4 Data-sharing test results

The bar graph in Figure 4-34 shows the ITR for an SAP Day Posting test run on Db2 13 and Db2 12 with the same user load and software and hardware configuration. It shows that Db2 13 has a comparable ITR to Db2 12. The system is tuned so that the response time is close to 1 second.

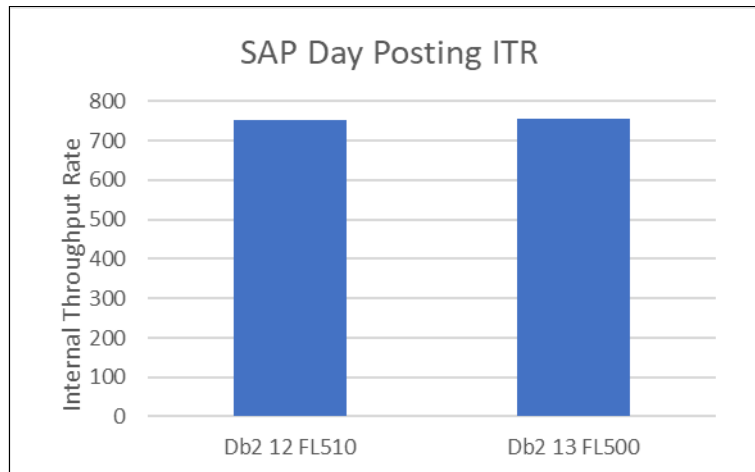


Figure 4-34 SAP Day Posting internal throughput rate

The z/OS LPAR CPU utilization and Db2 class 2 CPU in Table 4-8 shows that the difference between Db2 13 and Db2 12 is within noise level.

Table 4-8 SAP Day Posting CPU utilization and response time

Item	Db2 12 (FL 510)	Db2 13 (FL 500)
z/OS LPAR CPU utilization	81%	81%
Db2 class 2 CPU time (sec)	0.001996	0.001963

The SAP Account Settlements measurements were conducted by using Db2 12 at FL 510 and Db2 13 at FL 100 while processing all 100 million accounts. The results on both releases are similar regarding elapsed time and ITR, as shown in Figure 4-35.

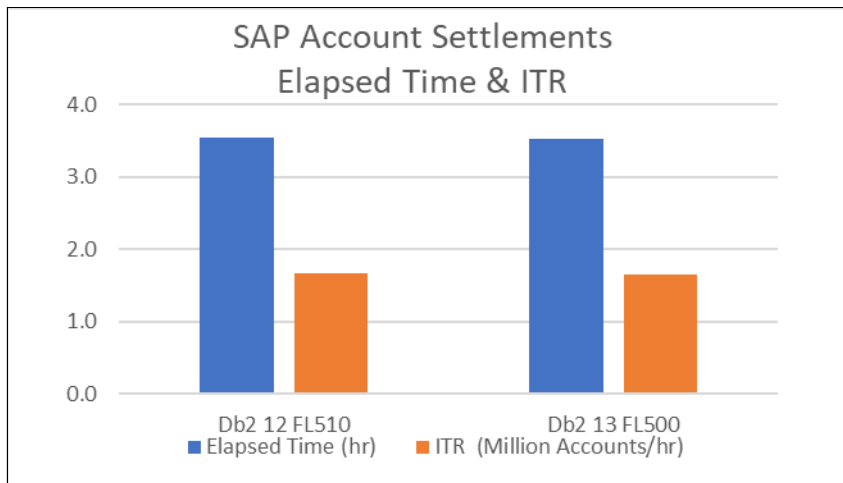


Figure 4-35 Account Settlement elapsed time and internal throughput rate

The z/OS LPAR CPU utilization and Db2 class 2 CPU time are also similar for the two versions during these runs, as shown in Table 4-9.

Table 4-9 SAP Account Settlement CPU time

Item	Db2 12 FL510	Db2 13 FL500
z/OS LPAR CPU utilization	85%	86%
Db2 class 2 CPU time (sec)	0.01601	0.01602

## 4.8.5 Conclusion

Db2 must perform well with business-critical SAP applications. Running SAP workloads with Db2 12 and Db2 13 shows that Db2 13 continues to perform well with respect to CPU time, elapsed time, throughput, and scalability.



## Data sharing

This chapter describes the following Db2 13 performance enhancements that are related to Db2 data sharing:

- ▶ Partition-by-range table space relative page numbering enhancements
- ▶ Group buffer pool cast-out-related changes in Db2 13
- ▶ Data sharing with distance

## 5.1 Partition-by-range table space relative page numbering enhancements

Db2 12 introduced a new attribute for partition-by-range (PBR) table spaces called *relative page numbering* (RPN), which features partition sizes up to 1 TB and less-disruptive partition management. PBR table spaces with RPN allow many improvements in space allocation. However, with the original design in a data-sharing environment, the algorithm that is used to generate the resource hash values for page P-locks can result in a high degree of false contentions in the lock structure in some cases, especially when running applications with row-level locking (RLL). This situation can result in high CPU usage in the *ssnmlRLM* address space and XCF address space. The applications might also see elongated suspension times in global contention.

To improve the performance and scalability of PBR RPN table spaces for applications that use RLL, the algorithm for generating the resource hash values for page P-locks was redesigned in Db2 13. With the new algorithm, the resource hash values are better distributed with higher levels of uniqueness, and false contentions are minimized.

**Important:** Applications that do not generate many page P-locks, such as applications that use PBR RPN table spaces with page-level locking, do not seem to have high false contention. For these applications, using PBR RPN table spaces does not cause a performance issue.

### 5.1.1 Requirements

This enhancement is available for Db2 13 data-sharing groups that are operating at function level (FL) 500 or later.

For PBR RPN table spaces that are created in Db2 12 and Db2 13 FL 100, the **REORG** utility with the **SHRLEVEL REFERENCE** or **CHANGE** option, or the **LOAD** utility with the **REPLACE** option must be run to use the new hash values. For more information, see 5.1.3, “Usage considerations” on page 193.

### 5.1.2 Performance measurement

To determine the effectiveness of the new hash algorithm for reducing false contention of PBR RPN table spaces with RLL, we designed a workload that randomly updates records in a table with RLL in a 2-way data-sharing environment and measured its performance on Db2 12 and Db2 13. The workload is created in such a way that it uses many page P-locks on data pages. The results show that the workload completes the same number of data updates in a much shorter elapsed time and uses much less CPU time in Db2 13.

The workload also ran with a PBR table space that was defined with absolute page numbering (APN). The results show that completing the same number of updates in Db2 13 by using a PBR RPN table space has similar performance as when running with a PBR APN table space.

In Chapter 4, “Workload-level measurements” on page 149, some of the high-insert workloads that use PBR RPN table spaces and RLL also see benefits from this enhancement with reduced false contention. For more information about how the new hash algorithm helps to improve their performance, see “Archive sequential insert” on page 162 and “Journal table insert” on page 166.

## Measurement environment

The performance measurements that were conducted used the following setup:

- ▶ IBM z16 hardware with eight general CPs and two IBM zSystems Integrated Information Processors (zIIPs)
- ▶ Two Coupling Facilities (CFs) at CF LEVEL 25, with three dedicated CPUs each
- ▶ 512 GB of logical partition (LPAR) memory
- ▶ 1 GB lock structure size
- ▶ z/OS 2.5
- ▶ Db2 13 FL 500
- ▶ Db2 12 FL 510

## Measurement scenario description

The workload is a multi-thread random update workload that has 10 threads running concurrently on each Db2 member of a two-way data-sharing configuration. The table space that the update threads access is a PBR table space with 40 partitions and contains 40,000,000 rows of data. The table space is defined by using **LOCKSIZE ROW** and **PAGENUM RELATIVE**, which means that it is using RLL and RPN. The update threads access the rows randomly through a unique index, and during the measurements, all the partitions are group buffer pool (GBP)-dependent. Each thread updates 200,000 rows before it completes. With 100 random updates done per commit, the workload is triggering 99.8 page P-locks on data pages per commit.

For the Db2 12 measurement, the table space was created in FL 510 and populated at the same FL. This measurement is the base one, and the PBR RPN table space uses the old hash algorithm for page P-locks.

For the Db2 13 measurement, the table space was created and populated in FL 500. The PBR RPN table space uses the new hash algorithm for page P-locks.

To understand how the workload performs with a PBR RPN table space compared to a PBR table space that used absolute page numbering (APN) in Db2 13, we also ran another Db2 13 measurement with a PBR APN table space that is created in FL 500 and populated with the same data.

## Results

The measurement results are described in two parts:

- ▶ The first part covers observations of the workload performance when using a PBR RPN table space and comparing Db2 13 and Db2 12 measurements.
- ▶ The second part covers observations comparing the workload performance of Db2 13 measurements when using a PBR RPN table space versus a PBR APN table space with the same data in the table space.

Comparing performance between Db2 versions when using a PBR RPN table space, the Db2 13 measurement of the random update workload shows the following improvements over Db2 12 regarding reductions in CPU time, as shown in Figure 5-1:

- ▶ Total Db2 CPU usage including zIIP time, which is calculated as the sum of total Db2 address space CPU and class 2 CPU time, is reduced by 51%.
- ▶ *ssnm*IRLM address space CPU usage is reduced over 99%.
- ▶ XCFAS address space CPU usage is reduced over 98%.
- ▶ Total LPAR CPU usage is reduced by 58%.

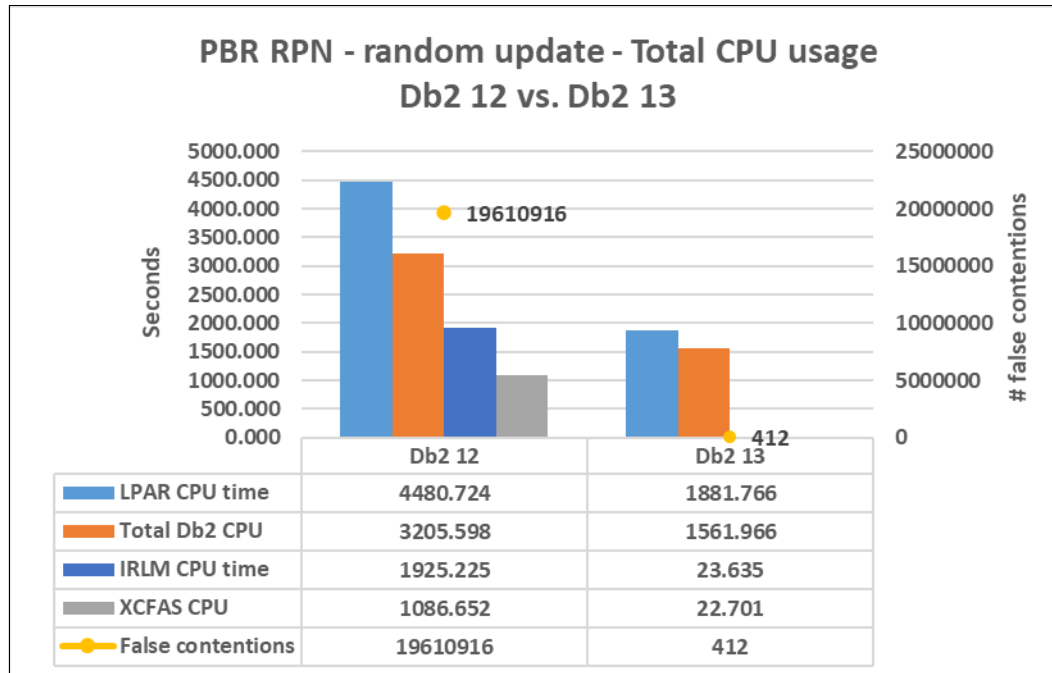


Figure 5-1 PBR RPN Db2 12 versus 13 random update workload CPU usage of LPAR, Db2, and XCF

We also observed the following reductions in elapsed time and global contention suspension time, as shown in Figure 5-2 on page 191:

- ▶ Total class 2 elapsed time of the workload is reduced by 77%.
- ▶ Total class 3 global contention suspension time, and the number of suspension events, is reduced by 99%.

All these improvements result from the vast reduction in the number of false contention suspensions, from 19,610,916 to a mere 412.

An increase of 20% was observed in class 2 CPU usage, as shown in Figure 5-2 on page 191, and an increase of *ssnm*DBM1 IIP System Recovery Boost (SRB) time of 98% was observed when the workload was run under Db2 13 (Db2 12 with 134.276 seconds and Db2 13 with 259.319 seconds). However, the class 2 CPU and *ssnm*DBM1 IIP SRB time increases are justifiable tradeoffs for greater reduction in the total Db2 CPU usage and to achieve a 4.7 times increase in the update rate compared to the Db2 12 measurement.

**Note:** The workload is designed so that it has many page P-locks on data pages, so it performed poorly in Db2 12 when running with PBR RPN table space by using RLL. Other workloads, such as some of the high-insert workloads that also use PBR RPN table spaces with RLL, have much lower false contention rates (below 0.3%), and the impact of false contentions are much less notable for them.

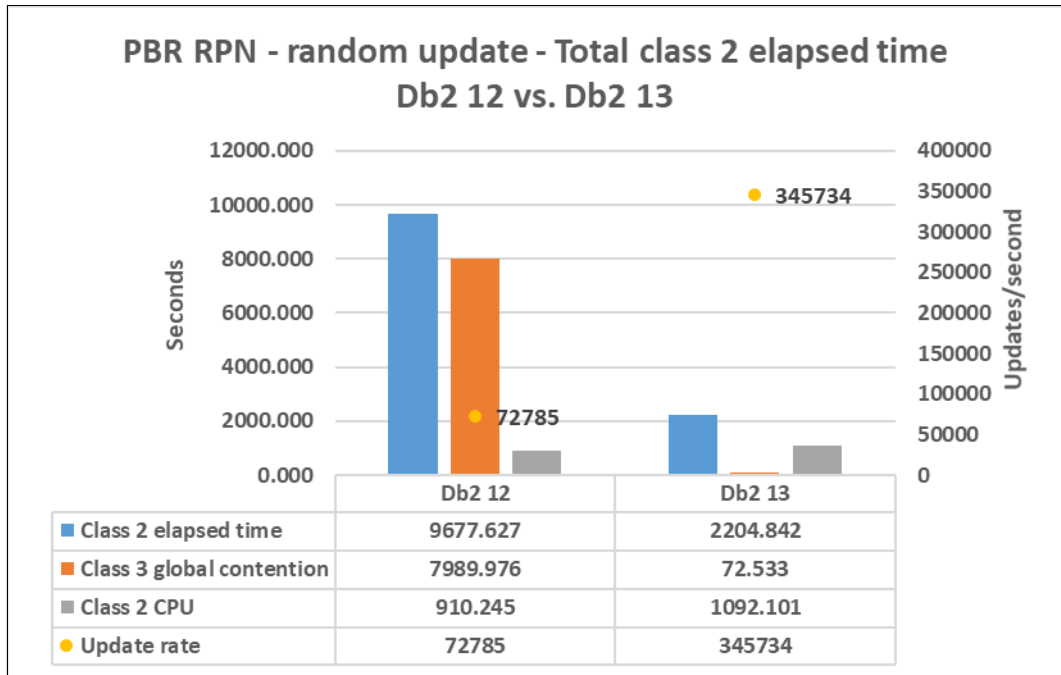


Figure 5-2 PBR RPN Db2 12 versus Db2 13 random update workload key elapsed time metrics

Comparing the performance of the random update workload between using PBR with RPN versus APN in Db2 13, there is no CPU or elapsed time regression. All the key performance metrics such as class 2 CPU time, elapsed time, Db2 address space CPU usage, and update rates had differences within a range of -1% - 1%.

**Note:** The workload also ran by using a Db2 12 data-sharing group with a PBR APN table space, and the performance is like the Db2 13 PBR APN measurement.

Figure 5-3 and Figure 5-4 illustrate the performance numbers and comparisons between the Db2 13 PBR APN and RPN tests.

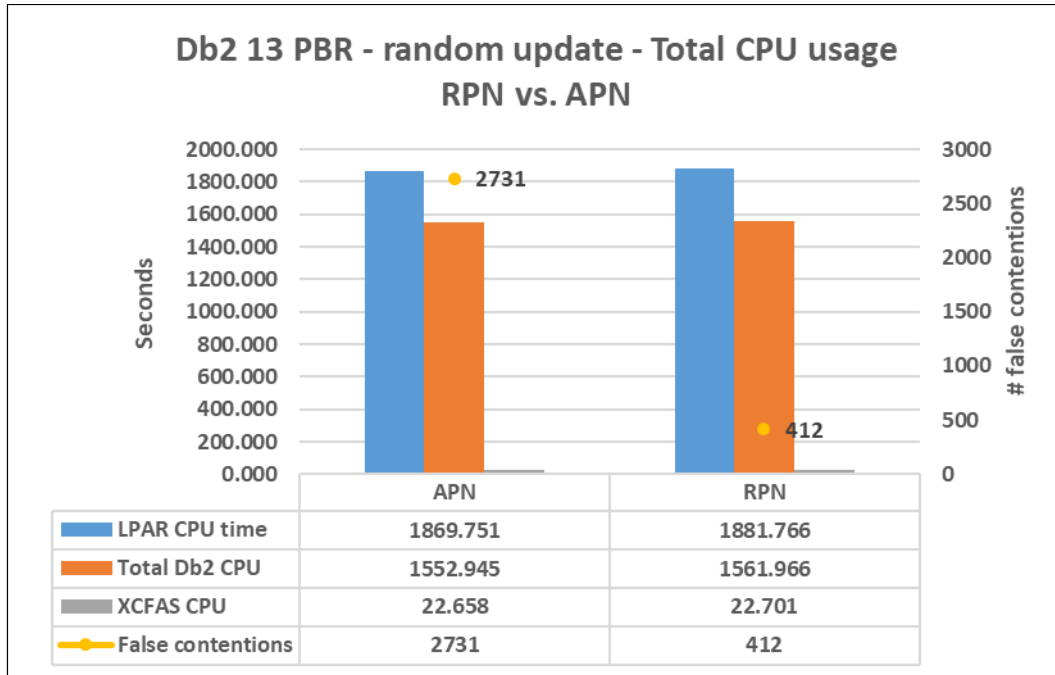


Figure 5-3 Db2 13 PBR APN and RPN random update workload CPU usage of LPAR, Db2, and XCF

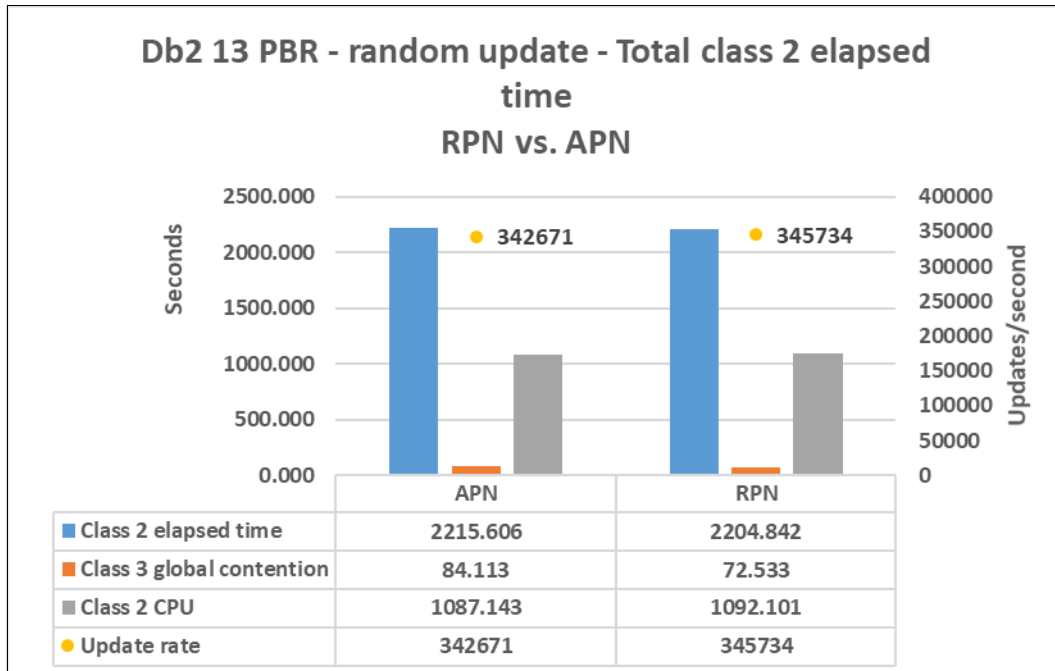


Figure 5-4 Db2 13 PBR APN versus RPN random update workload key elapsed time metrics



## Summary of results

The performance tests show that Db2 13 improves scalability and performance capabilities over Db2 12 for PBR table spaces with RPN when using RLL. PBR RPN performance is not negatively affected when using page-level locking in Db2 12. Also, if you have PBR RPN table spaces in your workload in Db2 12 and do not see high false contentions, do not expect significant improvements by converting the table spaces to use the new page P-lock hash values.

### 5.1.3 Usage considerations

The lowest FL for the implementation of the new page P-lock hash algorithm is V13R1M500, which indicates that all the members of a data-sharing group are running in Db2 13.

All new PBR table spaces that are created in Db2 13 with FL 500 and later with **PAGENUM RELATIVE** will be formatted to support the new hash algorithm.

Existing PBR table spaces that were created before FL V13R1M500 with **PAGENUM RELATIVE** do not support the new hash algorithm automatically. A conversion is needed to format the table space header pages with the required settings.

After you are at FL V13R1M500 and later, there are two ways to convert an existing PBR table space or individual partitions to use the new hash algorithm for RPN:

- ▶ Run the **REORG** utility with **SHRLEVEL REFERENCE** or **CHANGE**.
- ▶ Run the **LOAD** utility with the **REPLACE** option.

To see whether you can benefit from this enhancement, complete the following steps:

1. If you have PBR RPN table spaces that were created with Db2 12 or Db2 13 before FL500 in your workloads, check whether they were created with **LOCKSIZE ROW**.
2. Check the DATA SHARING section of the IBM OMEGAMON for Db2 Performance Expert (OMPE) DB2PM accounting reports of the applications that access these table spaces.
3. If you see a high number of P-lock requests, and the number of FALSE CONTENTIONS is greater than the number of SUSPENDS – IRLM (global internal resource lock manager (IRLM) contentions), consider converting these PBR RPN table spaces to use the new page P-lock hash values after you migrate to Db2 13 FL 500. The conversions should allow your applications to run with fewer false contentions.

### 5.1.4 Conclusion

If you are considering leveraging the PBR table spaces with RPN for their traits, such as a single partition growing up to 1 TB, or if you are planning to convert your existing partition-by-growth (PBG) table spaces to PBR, there is no better time to do it than with Db2 13. After activating FL V13R1M500, you can convert PBG table spaces into PBR RPN table spaces with minimal application impact, as described in [Converting tables from growth-based to range-based partitions](#). With the performance improvement for PBR RPN table spaces that are described in this section, you can expect your workloads to run with optimum performance by using PBR RPN table spaces.

## 5.2 Group buffer pool cast-out-related changes in Db2 13

Db2 workload sizes in customer installations have grown over the years. As a result, GBP CF structure sizes are getting larger. With new generations of IBM zSystems hardware and disk systems providing faster processing and I/O speeds, GBP cast-out operations can write the same number of data pages to disk faster. With these larger GBP sizes and faster cast-out speeds, some of the code design that affected GBP cast-out activity became non-optimal.

In Db2 13, the frequency at which Db2 evaluates GBP cast-out threshold (GBPOOLT) has been reduced to 0.2 of the original design. The purpose of this change is to trigger GBP cast-out actions more frequently. This change can improve transaction performance by ensuring that enough storage is available in the GBP with more efficient cast outs.

Another change in Db2 13 that is related to GBP cast out is reducing the time that a transaction waits after encountering a GBP write failure condition before resuming and retrying the GBP write. The wait time allows GBP cast-out operations to free enough storage in the GBP. As GBP cast-out efficiency improves, storage can be freed in the GBP structure more quickly than before. Reducing wait time (from 1 second in Db2 12 to 100 milliseconds in Db2 13) after the transaction experiences a GBP write failure helps shorten the transaction's update commit time because the transaction can retry the failed GBP write more quickly.

### 5.2.1 Requirements

This enhancement is available for Db2 13 systems that are operating at FL 100 or later.

### 5.2.2 Performance measurements

To make sure the two Db2 13 GBP cast-out enhancements achieve the intended performance improvements, we designed a workload that runs in a two-way data-sharing environment. The workload triggers heavy GBP write activity and stresses the GBP storage enough to experience some GBP write failures. We measured the workload performance in Db2 12 and in Db2 13. Comparing the performance numbers of the measurements, we observed reduced Db2 class 2 elapsed time, update commit time, and fewer GBP write failures for the workload when run with Db2 13 code.

#### Measurement environment

The performance measurements that were conducted used the following setup:

- ▶ IBM z15 hardware with eight general CPs and two zIIPs on each LPAR
- ▶ 512 GB memory on each LPAR
- ▶ 2 CFs at CF LEVEL 24, each with three dedicated CPUs
- ▶ z/OS 2.4
- ▶ Db2 12 FL 510 with 2-way data-sharing
- ▶ Db2 13 FL 500 with 2-way data-sharing

## Measurement scenario description

The workload is a multi-thread random update and select batch workload that has 20 threads running concurrently on each Db2 member of a two-way data-sharing configuration. Each thread updates 500,000 rows and selects 1,500,000 rows randomly through a non-GBP-dependent unique index before completing. The table space that the workload accesses is GBP-dependent and defined to a GBP with a CF structure size of 2 GB. The GBP CF structure has duplexing enabled. The GBPOOLT for the GBP is set to the default value of 30%. The directory-to-data ratio of the GBP is 10 during both measurements. The GBP checkpoint interval is set to 4 minutes.

We tuned the workload so that GBP cast out is triggered frequently in Db2 12.

## Results

With the cast-out changes in Db2 13, we see solid improvements of the workload performance. Comparing the performance numbers of the Db2 13 measurement to the Db2 12 measurement, we see the following key results:

- ▶ The GBP cast-out threshold is triggered more frequently. In Db2 12, it is triggered 36 times, and in Db2 13, it is triggered 113 times.
- ▶ The number of pages being cast out for the workload increases about 2%, which is a slight increase that was expected. Some pages might have been updated more times in Db2 12 than in Db2 13 before being cast out because cast outs are happening less frequently.
- ▶ Because cast outs are being triggered more frequently and with a similar number of pages being cast out, the *ssnmDBM1* address space shows a 25% IIP SRB time savings because of the improved cast-out efficiency.
- ▶ 34 GBP write failures were recorded in the Db2 13 test, and 87 were recorded in the Db2 12 test.
- ▶ Update commit time for the application is reduced by 28% because of the shorter transaction suspension time when GBP write failures happen and also because there are fewer GBP write failures.
- ▶ Class 2 elapsed time is shortened by 14%, mainly because of the update commit time improvement.
- ▶ Class 2 CPU time has a slight increase of 0.7%, which is within the measurement noise level.

We see 15% more synchronous database I/Os, which is the result of more SYN.READ(XI)-NO DATA RETURN in the Db2 13 measurement. This increase occurred because the update rate is higher in Db2 13 and the data in GBP is being replaced at a quicker pace, so there is a greater chance of data not being returned from the GBP.

Figure 5-5 and Figure 5-6 illustrate these key performance results.

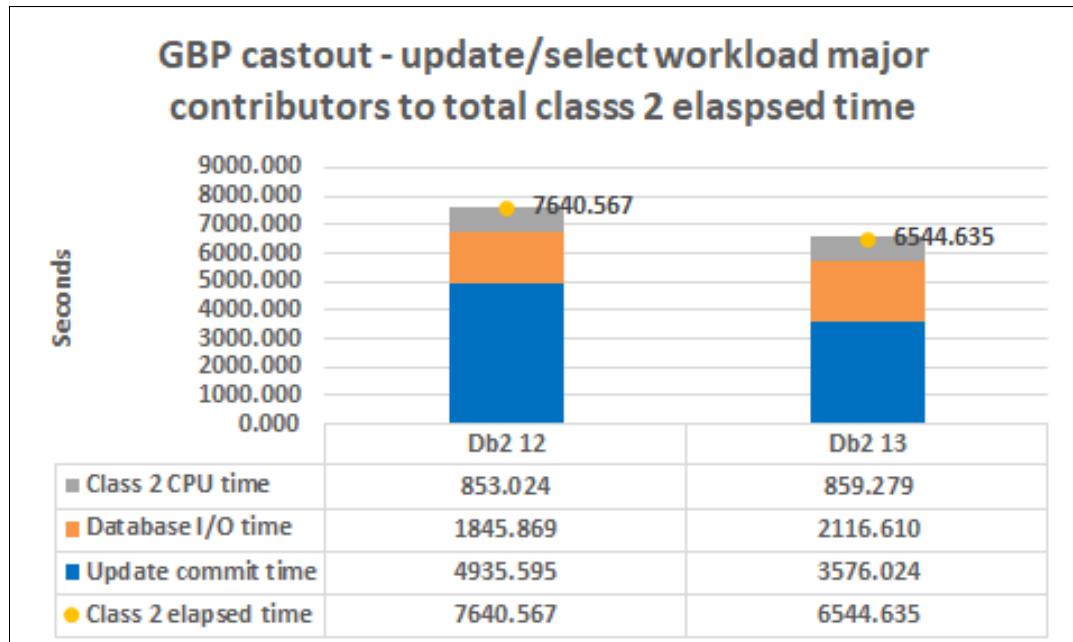


Figure 5-5 Random update and select workload major contributors to total Class 2 elapsed time

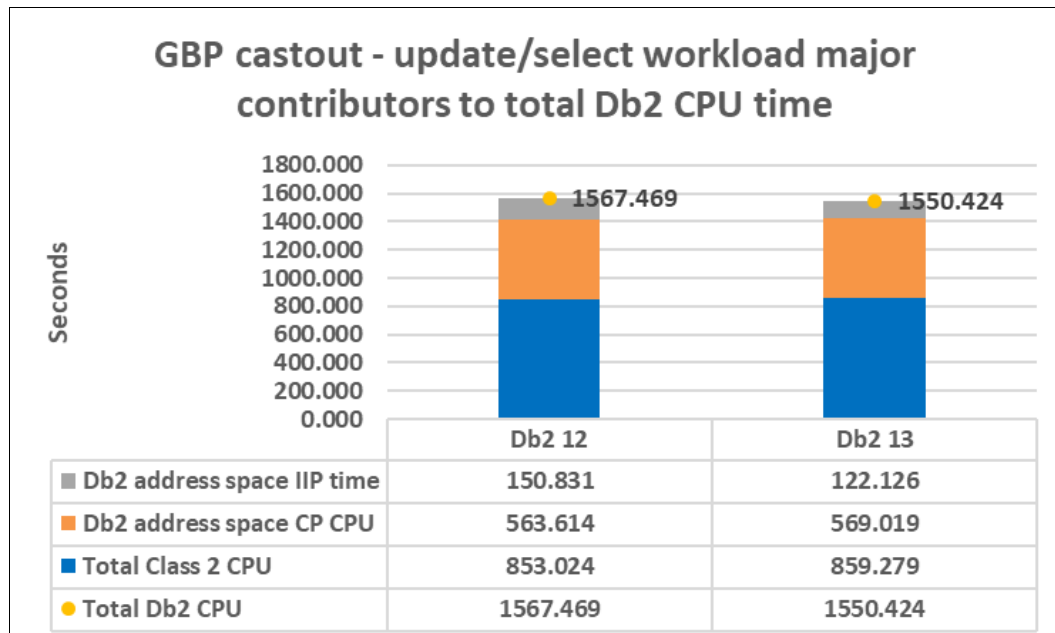


Figure 5-6 Random update and select workload major contributors to total Db2 CPU time

## Summary of the results

The performance tests that are described in this section show that the Db2 13 changes to perform GBP cast-out threshold checking more frequently can improve GBP cast-out efficiency. Reducing the transaction suspend time after encountering GBP write failures can improve application response time when GBP structure storage is under stress.

## 5.3 Data sharing with distance

A sysplex with a significant distance between the LPARs that hosts the Db2 members and CFs that host the CF structures is becoming more prevalent in the Db2 customer community. Many customers have questions about the performance of such a sysplex configuration.

To answer these questions, we acquired more equipment to simulate a sysplex with a distance up to 49 km. While this environment was being constructed, we conducted a series of experiments to investigate the intricacies of Db2 data sharing over a great distance. This section describes the insights that we gained from this exercise.

This section is not related to any of the new features of Db2 13. However, because this performance information is of interest to customers who are using or are considering using data sharing over a distance, we decided to add this information.

### 5.3.1 Requirements

There are no specific requirements for data sharing in a sysplex with distance.

### 5.3.2 Performance measurement

This section describes the different environment configurations that were used for our testing, the test scenarios that we used, and the results that we observed for the different environment configurations.

#### Measurement environment

A sysplex, which is composed of two MVS LPARs and three CFs, is used to run a two-way Db2 data-sharing workload.

Initially, all these components are in close vicinity. One CF, which is the focus of this study, is connected to both MVS LPARs through coupling over InfiniBand (CIB) links. The Db2 lock structure, shared communications area (SCA) structure, and primary GBPs are in this CF. This configuration serves as the baseline for this exercise.

In our testing, we increased the distance between one MVS LPAR and the CF by completing the following steps:

1. We replaced CIB links with Long Reach Coupling (CL5) links.
2. We configured CL5 links plus dense wavelength division multiplexing (DWDM) equipment with zero distance.
3. We configured CL5 links plus DWDM equipment with 25 km distance.
4. We configured CL5 links plus DWDM equipment with a 49 km distance.

All processors, including MVS LPARs and CFs, were IBM z14 processors. The CF link types that were used during the tests were CIB, CL5, and DWDM with distances of 0, 25, and 49 km.

The IBM brokerage transaction workload was used for this study. It ran in a Db2 12 data-sharing environment. For each of these configurations, a measurement was conducted to study the performance of this workload to better understand the performance characteristics of data-sharing workload in a sysplex over a distance.

## Measurement scenario description

Seven different configurations were tested to better understand the performance characteristics of Db2 data sharing over a distance. These seven configurations are described in the following subsections.

The CF links were the focus of this study, and the goal was to understand how the CF service time, which is a critical factor in data-sharing workload performance, varied in these different CF link configurations.

### Configuration A: No distance with CIB

This configuration consisted of the following components:

- ▶ MVS LPAR STLAB1A (Db2 CEB1) ← CIB x 4 → CF ICF0AZE1.
- ▶ MVS LPAR STLAB1D (Db2 CEB2) ← CIB x 4 → CF ICF0AZE1.
- ▶ CF ICF0AZE1 (\_LOCK1 simplex, \_SCA simplex, and \_GBP primary).
- ▶ CF ICF0B (\_GBP secondary).
- ▶ CF ICF0A did not hold any Db2 structures.

A schematic diagram of configuration A is shown in Figure 5-7.

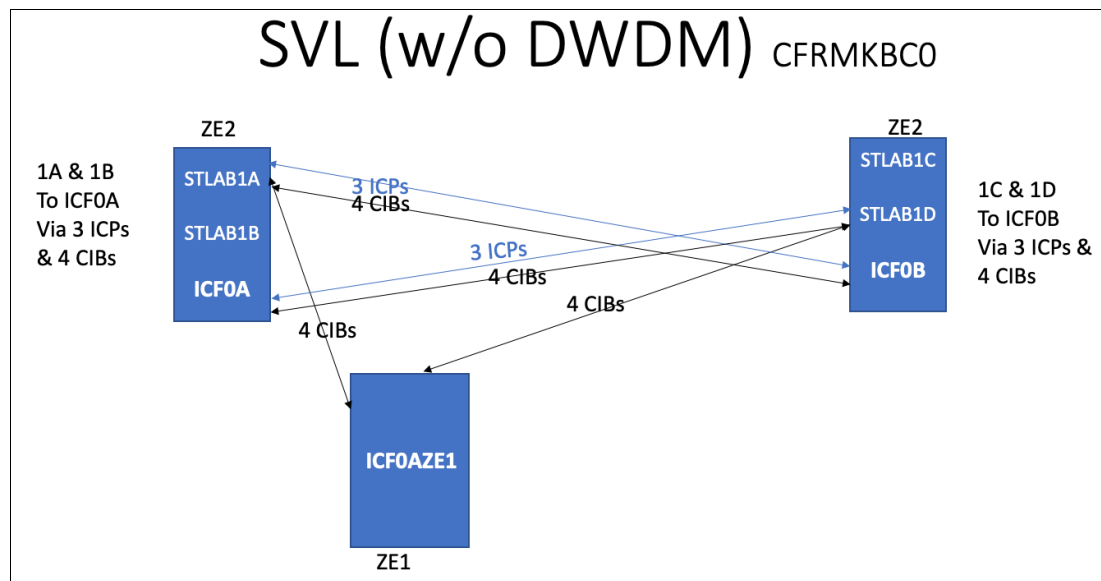


Figure 5-7 Schematic diagram of configuration A

### Configuration B: No distance with CL5

This configuration consisted of the following components:

- ▶ MVS LPAR STLAB1A (Db2 CEB1) ← CIB x 4 → CF ICF0AZE1.
- ▶ MVS LPAR STLAB1D (Db2 CEB2) ← CIB x 4 → CF ICF0AZE1.
- ▶ CF ICF0AZE1 (\_LOCK1 simplex, \_SCA simplex, and \_GBP primary).
- ▶ CF ICF0B (\_GBP secondary).
- ▶ CF ICF0A did not hold any Db2 structures.

A schematic diagram of configuration B is shown in Figure 5-8 on page 199.

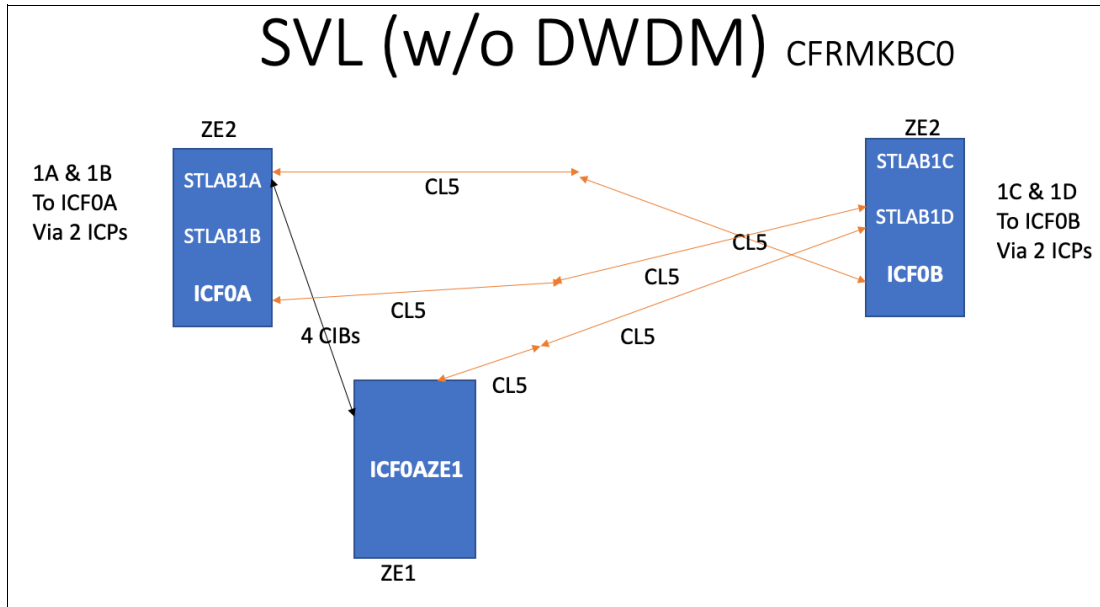


Figure 5-8 Schematic diagram of configuration B

**Configuration C: No distance with CL5 and DWDM installed**

This configuration consisted of the following components:

- ▶ MVS LPAR STLAB1A (Db2 CEB1) ← CIB x 4 → CF ICF0AZE1.
- ▶ MVS LPAR STLAB1D (Db2 CEB2) ← CL5 x 1 + DWDM with 0 km → CF ICF0AZE1.
- ▶ CF ICF0AZE1 (\_LOCK1 simplex, \_SCA simplex, and \_GBP primary).
- ▶ CF ICF0B (\_GBP secondary).
- ▶ CF ICF0A did not hold any Db2 structures.

A schematic diagram of configuration C is shown in Figure 5-9.

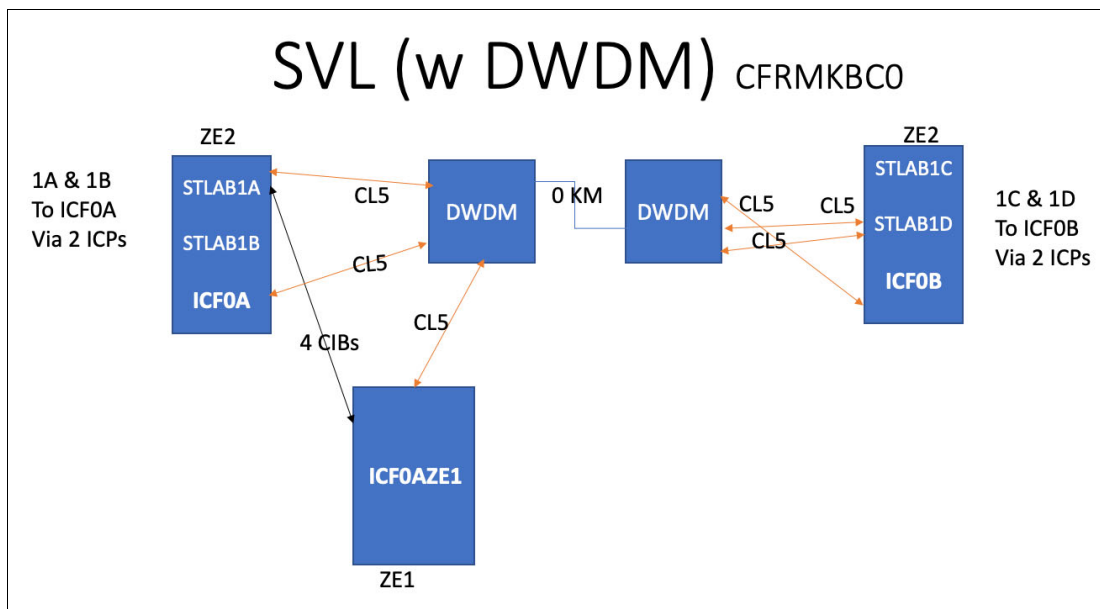


Figure 5-9 Schematic diagram of configuration C

**Configuration D: 25-km distance with CL5 and DWDM installed**

This configuration consisted of the following components:

- ▶ MVS LPAR STLAB1A (Db2 CEB1) ← CIB x 4 → CF ICF0AZE1.
- ▶ MVS LPAR STLAB1D (Db2 CEB2) ← CL5 x 1 + DWDM with 25 km → CF ICF0AZE1.
- ▶ CF ICF0AZE1 (\_LOCK1 simplex, \_SCA simplex, and \_GBP primary).
- ▶ CF ICF0B (\_GBP secondary).
- ▶ CF ICF0A did not hold any Db2 structures.

A schematic diagram of configuration D is shown in Figure 5-10.

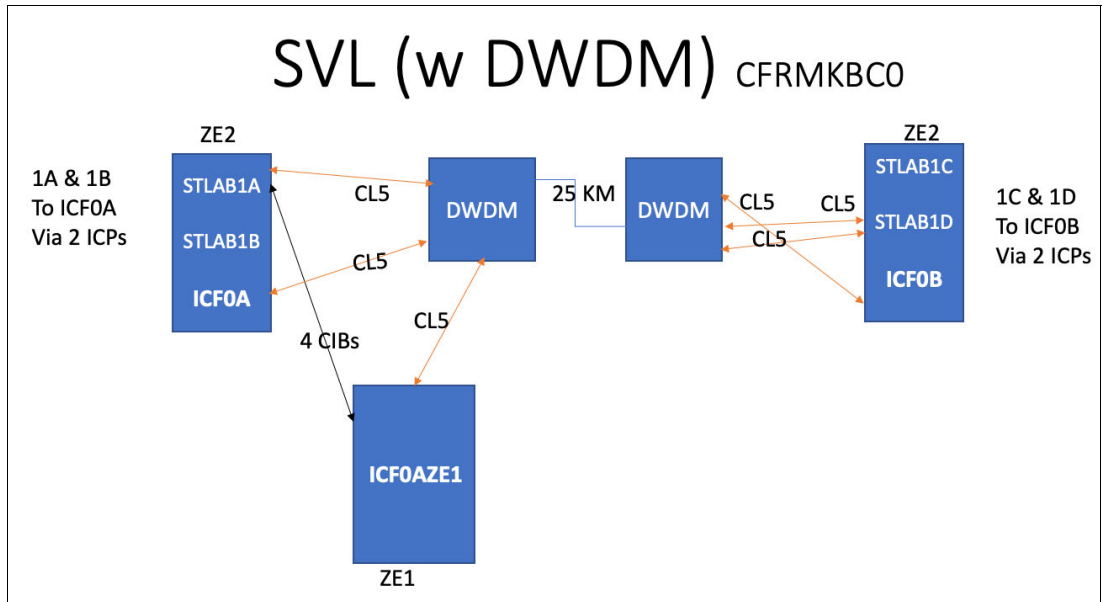


Figure 5-10 Schematic diagram of configuration D

**Configuration E: 49-km distance with CL5 and DWDM installed**

This configuration consisted of the following components:

- ▶ MVS LPAR STLAB1A (Db2 CEB1) ← CIB x 4 → CF ICF0AZE1.
- ▶ MVS LPAR STLAB1D (Db2 CEB2) ← CL5 x 1 + DWDM with 49 km → CF ICF0AZE1.
- ▶ CF ICF0AZE1 (\_LOCK1 simplex, \_SCA simplex, and \_GBP primary).
- ▶ CF ICF0B (\_GBP secondary).
- ▶ CF ICF0A does not hold any Db2 structures.

A schematic diagram of configuration E is shown in Figure 5-11 on page 201.



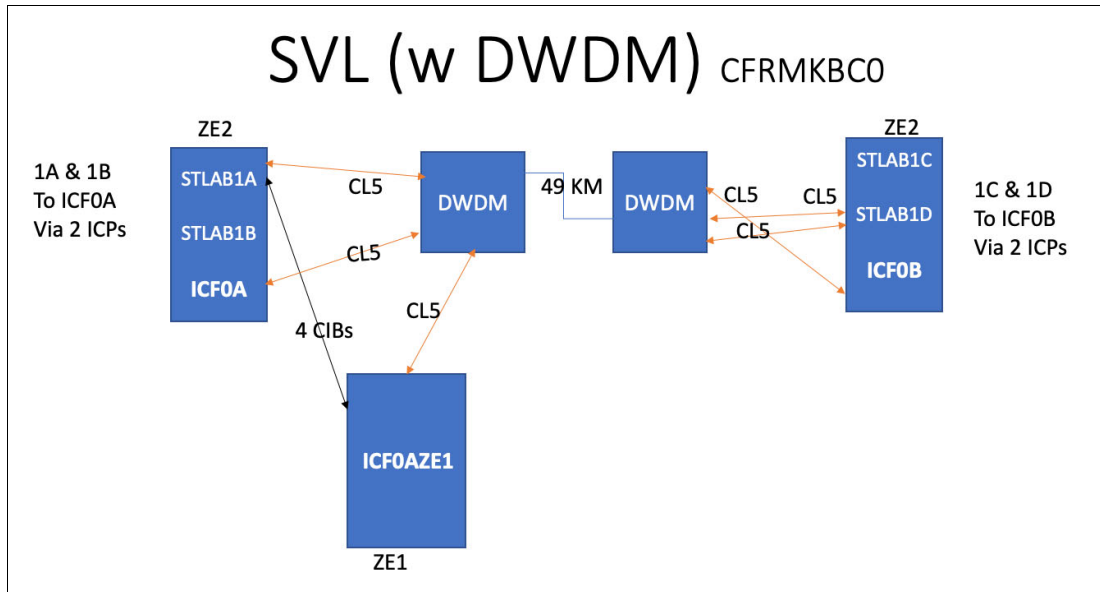


Figure 5-11 Schematic diagram of configuration E

**Configuration F: 25 km distance with two CL5s and DWDM installed**

This configuration consisted of the following components:

- ▶ MVS LPAR STLAB1A (Db2 CEB1) ← CIB x 6 → CF ICF0AZE1.
- ▶ MVS LPAR STLAB1D (Db2 CEB2) ← CL5 x 2 + DWDM with 25 km → CF ICF0AZE1.
- ▶ CF ICF0AZE1 (\_LOCK1 simplex, \_SCA simplex, and \_GBP primary).
- ▶ CF ICF0B (\_GBP secondary).
- ▶ CF ICF0A did not hold any Db2 structures.

This configuration was the same as configuration D, but with extra links that were defined.

A schematic diagram of configuration F is shown in Figure 5-12.

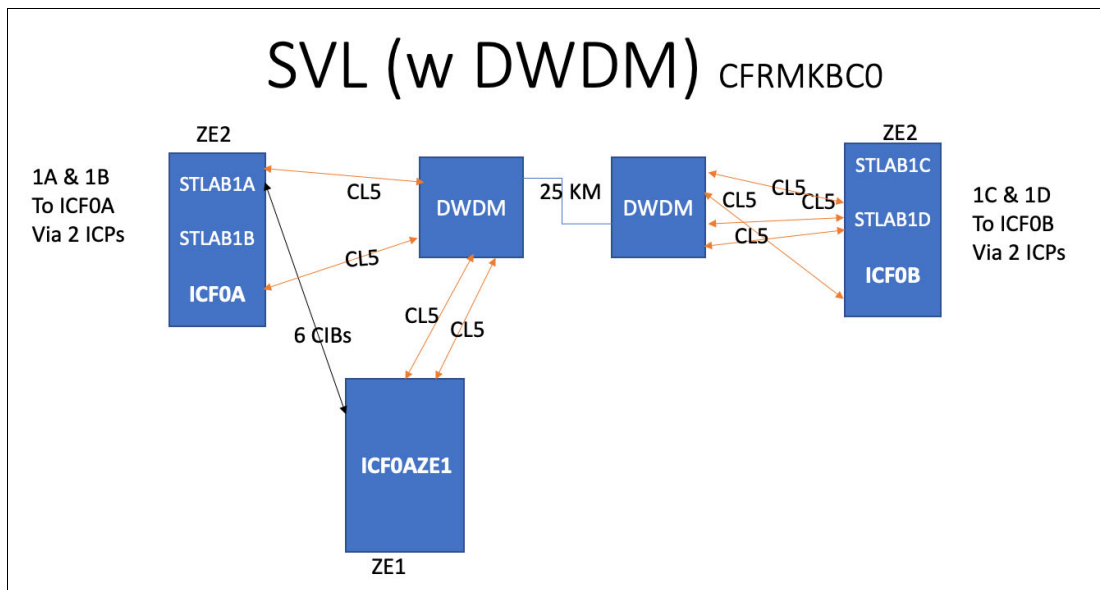


Figure 5-12 Schematic diagram of configuration F

**Configuration G: 49 km distance with two CL5 links and DWDM installed**

This configuration consists of the following components:

- ▶ MVS LPAR STLAB1A (Db2 CEB1) ← CIB x 6 → CF ICF0AZE1.
- ▶ MVS LPAR STLAB1D (Db2 CEB2) ← CL5 x 2 + DWDM with 49 km → CF ICF0AZE1.
- ▶ CF ICF0AZE1 (\_LOCK1 simplex, \_SCA simplex, and \_GBP primary).
- ▶ CF ICF0B (\_GBP secondary).
- ▶ CF ICF0A does not hold any Db2 structures.

This configuration was the same as configuration E, but with extra links that were defined.

A schematic diagram of configuration G is shown in Figure 5-13.

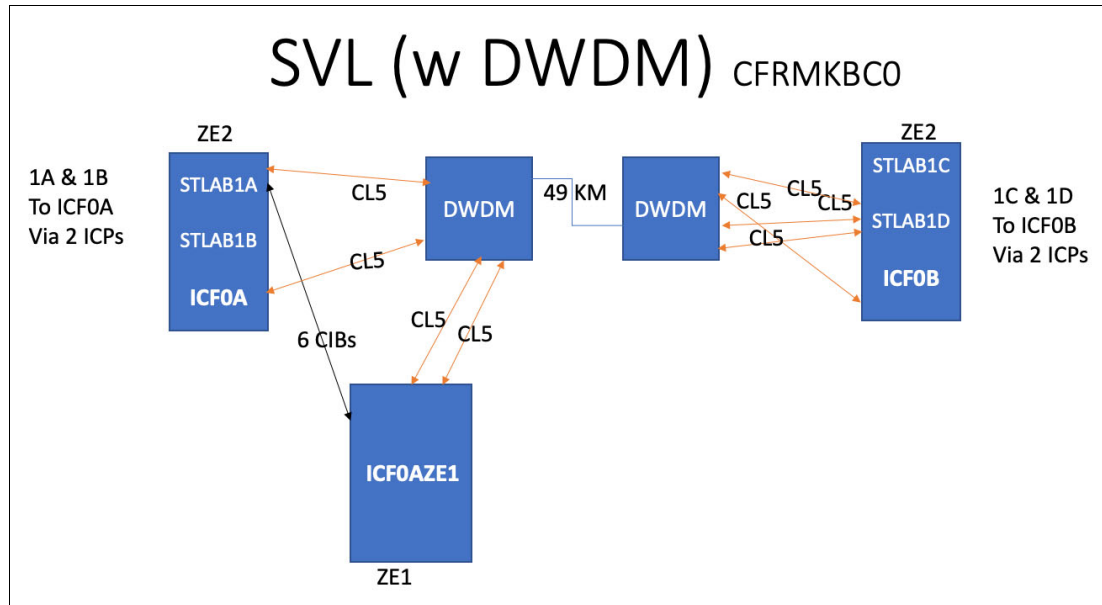


Figure 5-13 Schematic diagram of configuration G

**Measurement results**

The CF has three structure types: lock, cache, and list structures. Db2 data sharing employs all three types: \_LOCK1 (lock), \_GBPs (cache), and \_SCA (list).

From a performance perspective, \_LOCK1 and \_GBPs are more important than the \_SCA because the \_SCA structure is less frequently accessed during normal operation.

The average CF service times to the Db2 lock structure for configurations A - E are shown in Figure 5-14 on page 203.

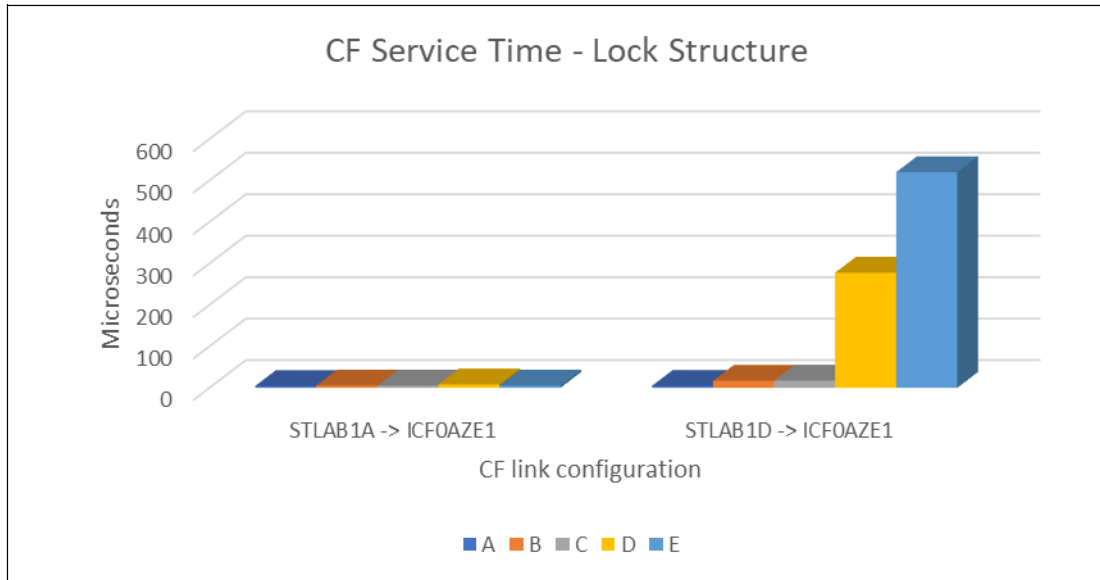


Figure 5-14 CF service time to `_LOCK1` structure for configurations A - E

The CF service time for `_LOCK1` is relatively stable between STLAB1A and ICF0AZE1 because four CIBs are used to connect them and no distance is involved here. However, between STLAB1D and ICF0AZE1, the CF service time varies from 5 ms in configuration A to 500 ms in configuration E, where MVS LPAR STLAB1D is 49 km away from CF ICF0AZE1.

The CF service time to Db2 cache structures is depicted in Figure 5-15.

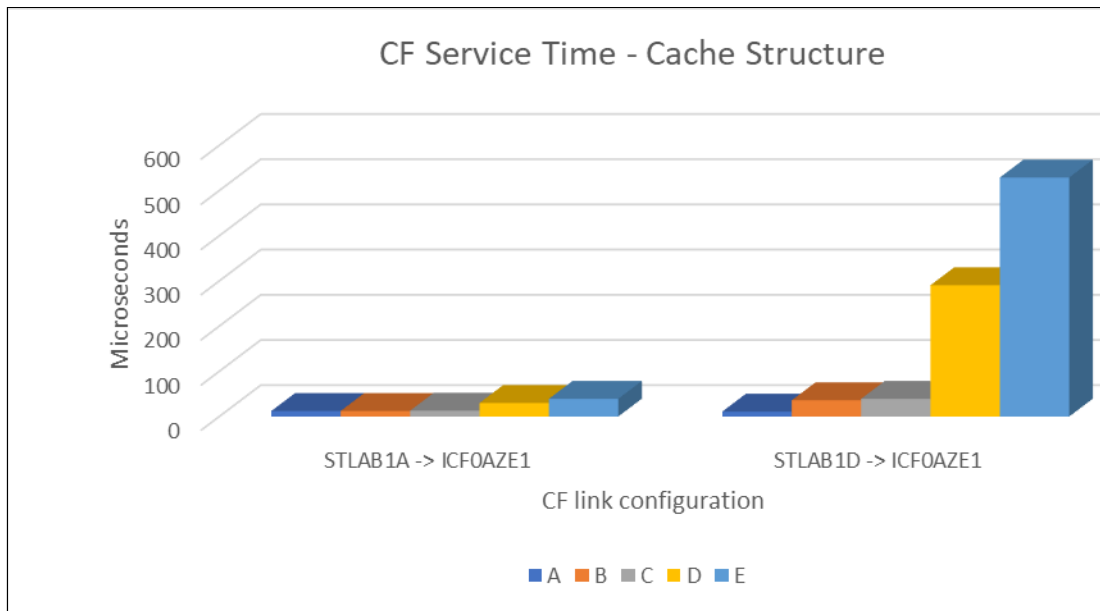


Figure 5-15 CF service time to `_GBPs` structure for configurations A - E

The CF service time for `_GBPs` is relatively stable between `STLAB1A` and `ICF0AZE1` from configurations A - C, but rises for configurations D and E for unknown reasons. The service time between `STLAB1D` and `ICF0AZE1` varies from the low tens of microseconds in configuration A to low 500s of microseconds in configuration E, where MVS LPAR `STLAB1D` is 49 km away from CF `ICF0AZE1`. The `_GBPs` here are 4 KB. Its service time is longer than `_LOCK1` because the data page itself also must be transferred in addition to directory information. By the same argument, `_GBPs` with 32 KB would have a longer CF service time than a 4 KB size page.

The CF service time of the Db2 list structure is depicted in Figure 5-16.

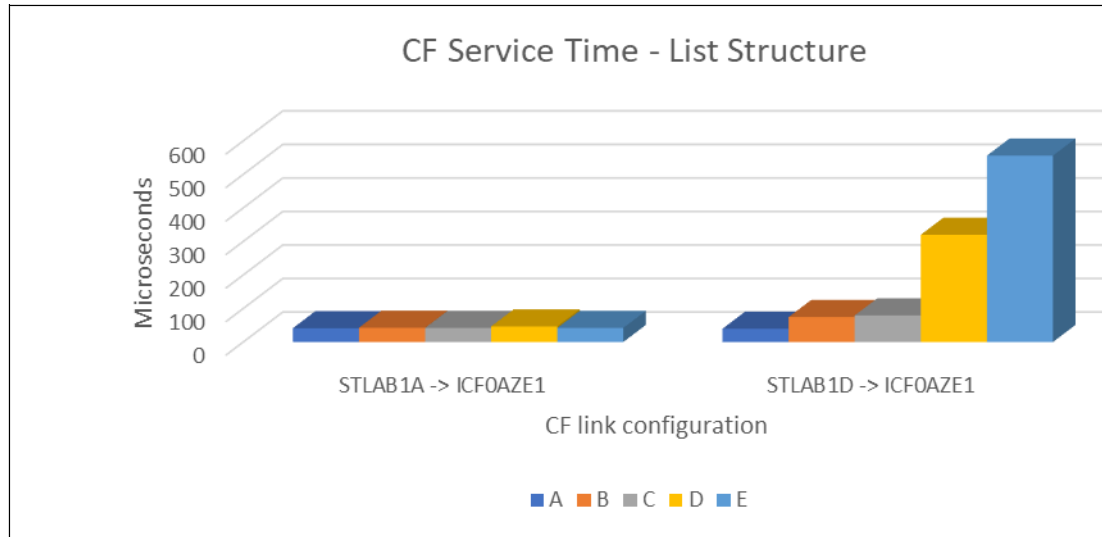


Figure 5-16 CF service time to `_SCA` structure for configurations A - E

The CF service time for `_SCA` is relatively stable between `STLAB1A` and `ICF0AZE1` from configurations A - E because 4 CIBs are connecting them in every configuration. The service time between `STLAB1D` and `ICF0AZE1` varies from 40 microseconds in configuration A to the mid-500s of microseconds in configuration E, where MVS LPAR `STLAB1D` is 49 km away from CF `ICF0AZE1`. Again, during a normal operation, access to the `_SCA` structure is infrequent compared to either `_LOCK1` or `_GBPs`. Therefore, a long CF service time does not significantly impact workload performance.

The overall CF service time for `STLAB1A` and `STLAB1D` respectively is shown in Figure 5-17.

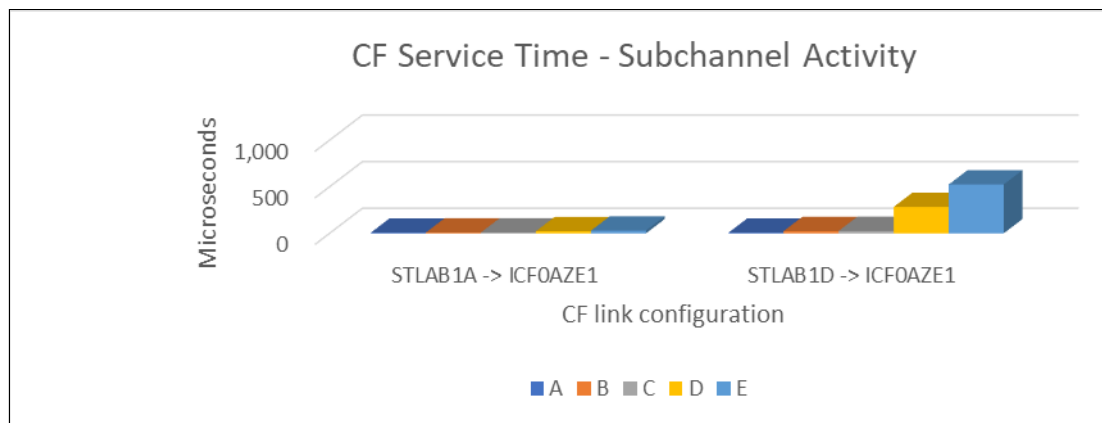


Figure 5-17 Overall CF service time for configurations A - E

The overall CF service time between STLAB1A and ICF0AZE1 is relatively stable from configurations A - C but rises for configurations D and E for reasons that are not understood at the time of writing. This behavior reflects what we observed for the \_GBPs discussed earlier.

As for the CF service time between STLAB1D and ICF0AZE1, it varies from single-digit microseconds in configuration A to low-500s microseconds in configuration E, where MVS LPAR STLAB1D is 49 km away from CF ICF0AZE1.

Due to the high CF service times in configurations D and E, the number of CF links were increased in an attempt to drive down CF service time for configurations D or E. Therefore, we created configurations F and G, which are depicted in Figure 5-11 on page 201 and Figure 5-12 on page 201.

Figure 5-18 and Figure 5-19 show the overall CF service time of configurations D and F and configurations E and G.

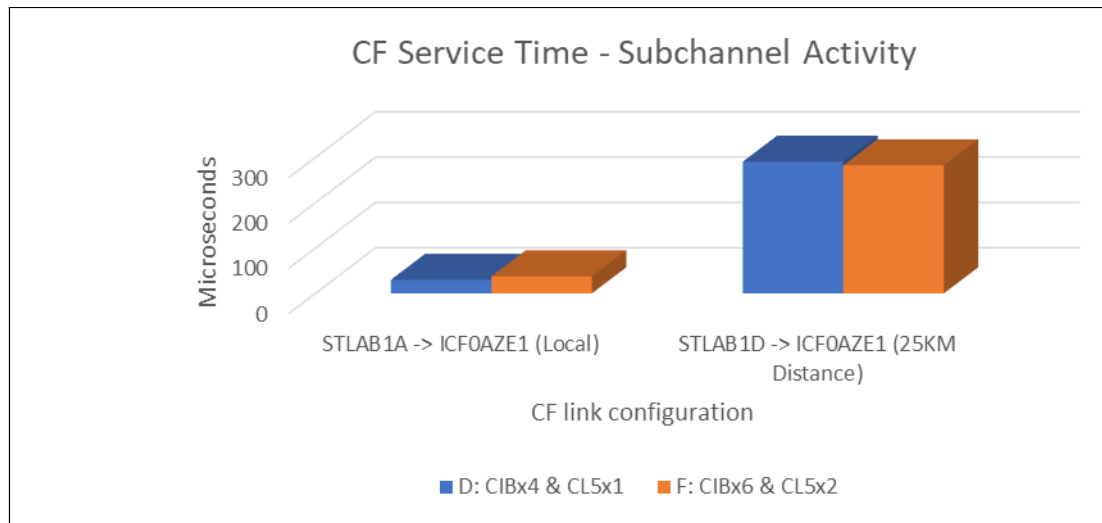


Figure 5-18 Overall CF service time for configurations D and F

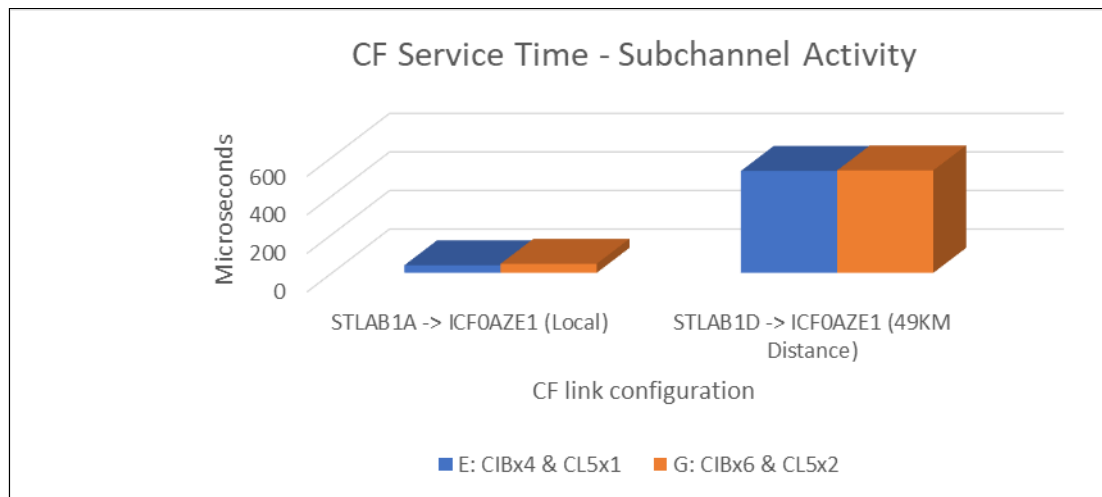


Figure 5-19 Overall CF service time for configurations E and G

It is evident that extra CF links do not help to reduce CF service times, especially from STLAB1A to ICF0AZE1. The CF activity report showed that the number of NO SUBCH conditions did go down considerably, but they did not result in shortened CF service times.

The additional CL5 link between STLAB1D and ICF0AZE1 stays neutral, with no improved or degraded performance. Therefore, it is not beneficial from a performance perspective to have two extra CL5 links.

Overall, CF service times from configurations A - E are reasonable. Every kilometer adds 10 ms of service time to each request. Therefore, an increase in CF service time is expected as distance goes up (speed of light remains unchanged).

Next, we look at how varying CF service times impacts Db2 transaction performance.

The IBM brokerage transaction workload that is running on Db2 12 is used for this study. Rather than looking at transaction performance at the data-sharing group level, we looked at the transactions that were run by each Db2 data-sharing member (or by the MVS LPAR because only one member is running on each LPAR).

Transaction class 2 CPU time for members CEB1 (on STLAB1A) and CEB2 (on STLAB1D) is shown in Figure 5-20.

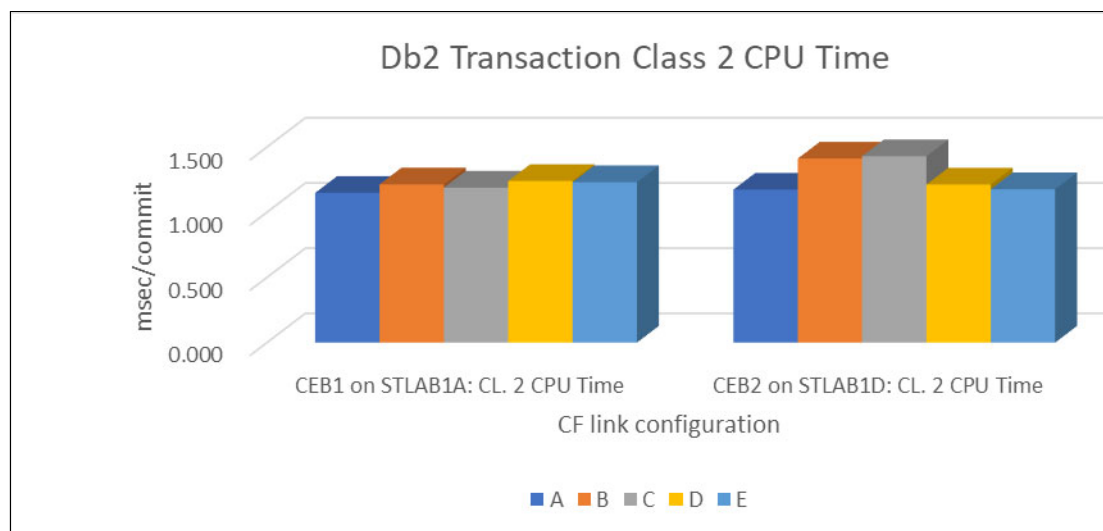


Figure 5-20 Class 2 CPU time of members CEB1 and CEB2 for configurations A - E

Transaction class 2 CPU time for configuration A is approximately the same between CEB1 and CEB2, that is, about a 2% difference. This result is expected because both STLAB1A and STLAB1D are connected to ICF0AZE1 with 4 CIBs.

For configuration B, transactions that run on STLAB1D use about 15% more class 2 CPU time than the ones on STLAB1A because the CF service time of STLAB1D is nearly twice of STLAB1A (see Figure 5-17 on page 204) even though the location (distance) is unchanged. The fact that requests that use CL5 links take longer than when they use CIB links explains this difference.

For configuration C, having DWDM equipment with no distance in between does not significantly alter CF service time compared to configuration B, which explains the similarity of class 2 CPU time between configurations B and C.

For configuration D, with 25 km distance between STLAB1D and ICF0AZE1, only 1% of the CF requests go synchronously, which tie up the processor. The remaining requests go asynchronously, which do not tie up the processor. Therefore, the elongated CF service time does not significantly impact CPU usage. As such, comparable class 2 CPU times between STLAB1A and STLAB1D are observed.

For configuration E, 49 km distance between STLAB1D and ICF0AZE1, only 0.6% of the CF requests go synchronously, which tie up the processor. This situation is similar to configuration D, except it is more pronounced. Our general finding is that comparable class 2 CPU times are observed between CEB1 and CEB2.

Transaction class 2 elapsed time for members CEB1 (on STLAB1A) and CEB2 (on STLAB1D) is shown in Figure 5-21.

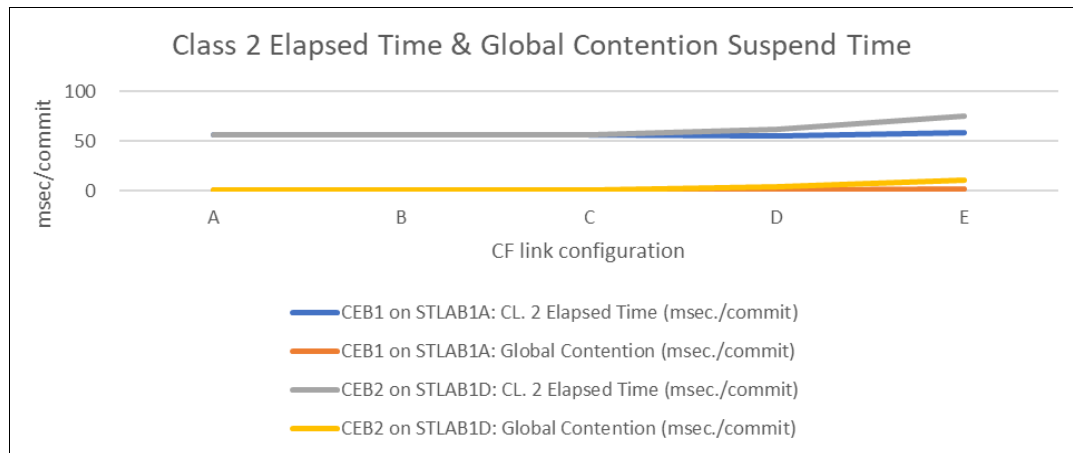


Figure 5-21 Class 2 elapsed time and global contention suspend time of members CEB1 and CEB2

Transaction class 2 elapsed time is approximately the same for configurations A, B, and C, but for configurations D and E, transaction elapsed time rises sharply for member CEB2, which is located 25 and 49 km away from CF ICF0AZE1. Analysis of the Class 3 suspension time indicates that this increase is largely driven by an increase in global contention suspend time due to the longer CF service time that is associated with distance. This suspension time is coming from CF sync requests being converted to asynchronous, not from actual resource contention. Therefore, long CF service time does not impact the CPU usage, being converted to asynchronous by z/OS. However, the long CF service time does affect transaction elapsed time.

Furthermore, global contention is closely related to IRLM processing, so more CPU time is consumed.

Figure 5-22 depicts IRLM CPU time per commit for Db2 systems CEB1 and CEB2 for configurations A - E.

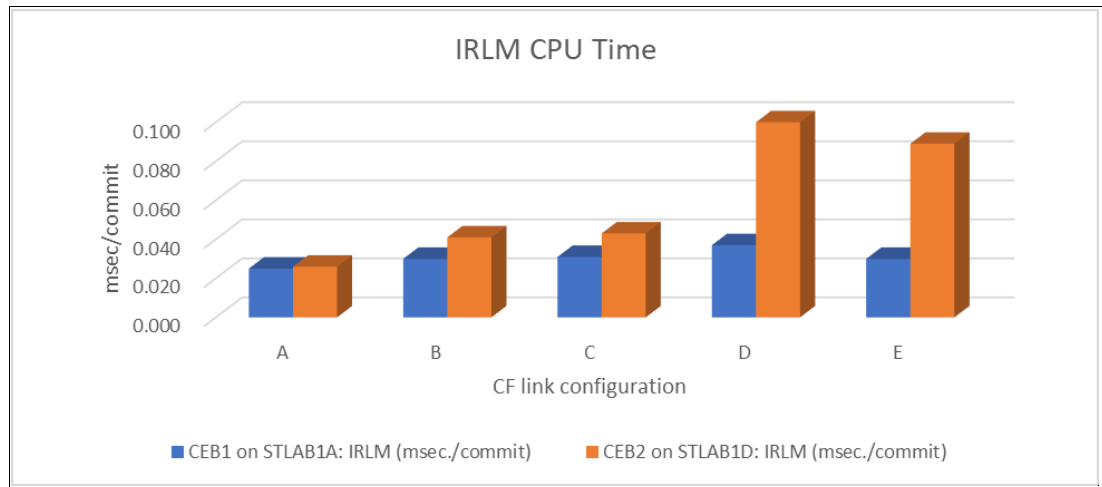


Figure 5-22 IRLM CPU time of members CEB1 and CEB2 for configurations A, B, C, D, and E

It is evident that IRLM CPU consumption is closely related to the CF service time or global contention suspend time that were observed in this study. IRLM CPU consumption is not zIIP eligible.

Internal throughput rate (ITR) is relevant to Online Transaction Processing (OLTP) workload performance. Again, rather than looking ITR at the data-sharing group level, this study looked at the ITR for each Db2 data-sharing member (or MVS LPAR).

Figure 5-23 shows workload ITR on STLAB1A and STLAB1D for configurations A - E.

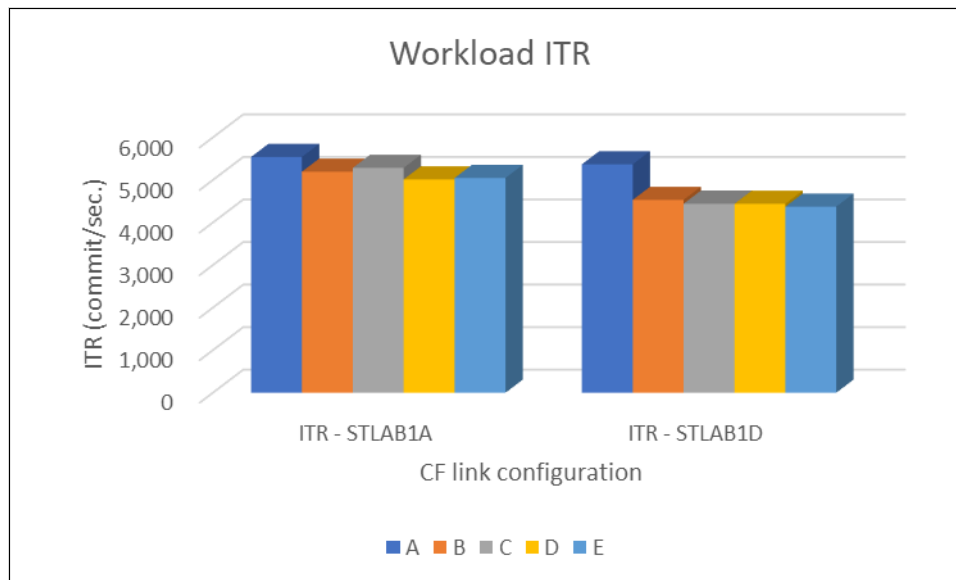


Figure 5-23 Workload ITR of STLAB1A and STLAB1D for configurations A - E



Workload ITR between STLAB1A and STLAB1D is comparable for configuration A. Workload ITR of STLAB1D is lower than the ITR of STLAB1A for configurations B - E. To sum up the observations in this study in general terms: the longer the CF service time, the lower workload ITR.

### 5.3.3 Monitoring information

RMF and Db2 accounting statistics traces were collected to analyze the performance characteristics of this workload, with a variable distance between one of the CFs and one of the Db2 members.

### 5.3.4 Conclusion

Measuring the performance of an OLTP workload running on a sysplex in which LPARs and CFs are initially within close vicinity but are gradually pulled away until 49 kilometers separate the LPAR and the CF where the Db2 `_LOCK1`, primary `_GBPs`, and `_SCA` structures are presented some unique challenges. However, the measurements that we obtained are valuable for planning configurations in which LPARs that host Db2 members are geographically distant from the CFs that host the CF structures.

In essence, CF link performance is studied here. CIB and CL5 types, when coupled with the distances of 0, 25, and 49 kilometers, allow us to have seven configurations, though two of them did not provide any additional insights from a performance perspective.

The type of CF link and distance between LPAR and CF determine the CF service time. Short CF service times allow CF requests to be done synchronously, which slightly increase CPU time. Longer CF service times drive CF requests asynchronously, keeping transaction class 2 CPU time the same. However, the longer CF service times also inevitably drive up global contention (suspension) time, which elongates Db2 class 2 elapsed time. The async conversion of the IRLM lock requests results in more suspend and resume activity by IRLM, and the consequence is increased IRLM CPU time. MVS (XES) decides to process CF requests synchronously or asynchronously, so Db2 cannot control it.

In summary, workload ITR is lower on the “remote” site as expected. Transaction class 2 CPU time is comparable on the remote site with a longer elapsed time, and the remote LPAR endures higher IRLM CPU time.





## SQL Data Insights

SQL Data Insights (SQL DI) is an optional feature of Db2 13 that brings artificial intelligence (AI) functions to the Db2 engine. SQL DI identifies valuable, hidden insight within a relational database for better business decisions. It reduces the time that is necessary to leverage AI. This feature does not require any extract-transform-load (ETL) processes to move the data off the IBM zSystems mainframe or hiring of experienced data scientists to design machine-learning (ML) models. Instead, by using the SQL DI user interface, you can add a Db2 table or view as an AI object, and then enable AI query functions and run semantic AI queries directly against the Db2 table or view to gain insight into the data and make business decisions.

After installing the SQL DI GUI, you identify the tables or views that you want to run AI queries against as AI objects, and then you enable AI query functions on the AI objects. Enabling AI query on an object trains an unsupervised neural network model, which is called *data embedding*, on the object and stores the generated ML model in the same Db2 system where the AI object is. After the model is created for the object and loaded into Db2, you can run semantic queries by using SQL DI built-in functions against the object. Currently, the following three SQL DI built-in functions are supported for semantic queries:

- ▶ The **AI\_SIMILARITY** function finds similar or dissimilar entities in a table or view.
- ▶ The **AI\_SEMANTIC\_CLUSTER** function checks what other entities exist in a table or view that could belong to a cluster of up to three entities.
- ▶ The **AI\_ANALOGY** function determines whether the relationship of a pair of entities applies to a second pair of entities.

Without any AI skills, you can use these Db2 SQL DI built-in functions in queries to infer hidden relationships between different entities in a table or view.

Using SQL Data Insights consists of the following two major steps:

- ▶ Enabling AI query
- ▶ Running an AI query

## 6.1 Enabling AI query

Enabling AI query on a table or view is the process that prepares the object for use with AI query functions. This step pre-processes the Db2 data, creates an ML model on the pre-processed data, and loads the model into Db2.

During the pre-processing phase, Db2 columns that are selected for the Enable AI Query process are converted to a textual format. Numeric data type values are clustered, and all numeric values are replaced with cluster identifiers. In the textual format of data, every numeric data type value is tagged with its column name and associated cluster identifier, and every categorical data type value is tagged with its column name.

During model training, an unsupervised model is trained on this textual representation of the Db2 data, and numeric vectors are generated for every unique value in the textual format data set. This set of unique values is referred to as *vocabulary* in the model. This neural network model, which is a collection of numeric vectors, is loaded into a Db2 table by using **ZLOAD**. The numeric vectors represent inter-column and intra-column semantic relationships between the different unique values in the AI object.

SQL DI has an embedded Spark cluster that enables AI Query. It uses a Spark cluster during the preprocessing phase while transforming Db2 data to textual format. The interface to train the model is provided by z/OS components, IBM Z Deep Neural Network Library (zDNN), IBM Z AI Optimization (zAIO), and IBM Z AI Data Embedding Library.

Besides all the Spark processing that is eligible to run on IBM zSystems Integrated Information Processors (zIIPs), numeric clustering during pre-processing and model training that does not run on Spark also are eligible to run on IBM zIIPs.

### 6.1.1 Requirements

To enable AI query processing, the requirements that are described in the following subsections must be in place.

#### Hardware requirements

An IBM z16, IBM z15, IBM z14, IBM z13, or IBM zEnterprise EC12 system.

#### Software requirements

- ▶ Db2 13 with function level (FL) 500 or later
- ▶ SQL Data Insights FMID HDBDD18
- ▶ z/OS 2.5 or 2.4:
  - zDNN:
    - For z/OS 2.5 with APARs OA62901
    - For z/OS 2.4 with APARs OA62849
  - IBM Z AI Optimization Library (zAIO):
    - For z/OS 2.5 OA62902
    - For z/OS 2.4 OA62886
  - IBM Z AI Data Embedding Library:
    - For z/OS 2.5 OA62903
    - For z/OS 2.4 OA62887

- ▶ z/OS Supervisor with APAR OA62728 for both z/OS 2.5 and 2.4
- ▶ IBM OpenBLAS with the following APARS:
  - PH44479 (for both z/OS 2.5 and 2.4)
  - PH45672 (for z/OS 2.5)
  - PH45663 (for z/OS 2.4)
- ▶ z/OS OpenSSH
- ▶ IBM 64-bit SDK for z/OS Java Technology Edition Version 8 SR7 FP11 or later

## 6.1.2 Performance measurement

A number of performance measurements were conducted to evaluate the Enable AI Query process by using AI objects ranging from small to large amounts of Db2 data.

In all the performance measurements, the following terms that are referenced in this document have a specific meaning:

<b>AI object</b>	A Db2 table or view that is added as AI object on which Enable AI Query processing is performed.
<b>AI object text data</b>	An AI object that is transformed to a textual data format with selected columns of Enable AI Query processing. This data is the input to model training.
<b>Number of words in the training text file (AI object text data)</b>	All the values in every categorical and numeric column that are selected for model training in the AI object.
<b>Vocabulary</b>	The total of number of unique values in the selected categorical columns and the number of clusters that are associated with the selected numeric columns in the AI object for the Enable AI Query process.
<b>Model (Vector) table</b>	The output of model training, which is stored as numeric vectors. Each row in the table represents a numeric vector. There is a numeric vector for every word in the vocabulary.

### Measurement environment

The SQL DI database DSNAIDB2 is set up in storage group DSNAIDSG with large enough capacity volumes to store multiple, trained ML models.

Because it is important that the SQL Data Insight processes are assigned the correct priorities, a separate SQL DI workload was defined in z/OS Workload Manager (WLM) on the logical partition (LPAR) where the performance measurements ran. The service classes that were defined in the SQL DI workload include the SQL DI application address space and Spark address spaces that are used by SQL DI. The service class for the Spark address spaces was defined with a lower priority than the SQL DI application address space. The SQL DI application address space was defined with a lower priority than Db2 address spaces.

The measurement environment used the following hardware and software:

- ▶ Hardware: IBM z15
- ▶ CPUs: Eight CPs
- ▶ LPAR memory: 4 TB
- ▶ z/OS: 2.5

- ▶ Db2 version: Db2 13 FL 501
- ▶ Spark executor memory: 24 GB
- ▶ Spark driver memory: 24 GB
- ▶ Spark executor cores: Three
- ▶ Spark driver cores: Three
- ▶ Training threads: Eight

### **Measurement scenario description**

The following performance measurement scenarios were run to evaluate elapsed time, CPU, memory, and direct access storage device (DASD) storage consumption:

- ▶ The first set of measurements ran to enable AI query for different sizes of Db2 views with a varying number of rows, but the same number of columns. The goal of this measurement was to understand the impact on elapsed time, CPU usage, and system resources as we increased the number of rows.
- ▶ The second set of measurements ran to enable AI query on Db2 views with the same number of rows, but varying the number of columns to demonstrate the influence of the number of columns in determining the resource requirements for training.
- ▶ The third set of measurements ran to enable AI query on different sizes of Db2 tables and views with a different number of columns and rows to demonstrate that the different resources that were required to train the model are not dependent on one dimension of the size of the input data to training (AI object text data).
- ▶ The fourth set of measurements ran on two identical tables of the same size but ran concurrently.

In all the performance measurement scenarios that are outlined here, all the columns in the AI object were selected for Enable AI Query processing.

### **Results**

This section describes the results for the four measurement scenarios.

#### ***Scenario 1***

The first set of measurements ran to enable AI query by increasing the size of the AI object by expanding the number of rows from 500 K to 101 million and keeping the number of columns constant (29 columns). The number of unique values (vocabulary) is expected to increase as number of rows increase in this set of measurements. Table 6-1 on page 215 shows the resource consumption when the AI query is enabled on AI objects with the same number of columns but with a different number of rows. An overview of the measurement results is shown in Table 6-1 on page 215.

Table 6-1 Resource consumption of Enable AI Query with a different number of rows

Enable AI Query	VIEW_1	VIEW_2	VIEW_3	VIEW_4	VIEW_5	VIEW_6
Number of rows in the AI object <sup>a</sup>	501 K	2,107 K	4,614 K	10,818 K	45,892 K	101,396 K
Size of the AI object (MB)	105	442	969	2,272	9,637	21,293
Number of numeric and categorical columns that are selected in the AI object to Enable AI Query	14 / 15	14 / 15	14 / 15	14 / 15	14 / 15	14 / 15
Size of AI object text data (MB)	342	1,439	3,151	7,358	31,268	69,063
Number of words in the training text file (AI Object text data) <sup>b</sup>	15,022,020	63,207,270	138,406,920	342,540,150	1,376,762,610	3,041,876,340
Number of words in the vocabulary	272,271	678,943	1,725,724	4,085,253	4,413,530	8,198,566
Elapsed time to pre-process data (minutes)	0.92	2.27	4.12	8.73	37.02	81.13
Size of ZLOAD binary size (MB) <sup>c</sup>	895	2,232	5,672	13,428	14,507	26,949
Size of the model table (extra DASD storage used by Db2) (MB)	892	2,224	5,653	13,383	14,459	26,858
Elapsed time to convert data to vector and load the data to Db2 (minutes)	3.97	16.9	36.4	100.47	458.92	941.68
Total elapsed time <sup>d</sup>	4.89	19.17	40.52	109.20	494.94	1,022.81
All Db2 address spaces CPU service time (seconds)	4.997	15.968	35.754	85.472	283.894	539.400
Db2 zIIP eligible (%)	49.1%	53.5%	53.2%	53.6%	57.8%	57.8%
SQL DI WLM CPU service time (minutes)	31.70	130.39	275.93	725.00	3,275.77	5,591.02
SQL DI zIIP eligible (%)	99.5%	99.9%	99.9%	100%	100%	100%

Enable AI Query	VIEW_1	VIEW_2	VIEW_3	VIEW_4	VIEW_5	VIEW_6
Maximum memory that is used by Enable AI Query (MB)	7,366	11,029	18,389	32,540	76,496	127,912
Extra DASD storage that is used by zFS (MB)	1,257	3,755	8,990	21,164	47,973	100,873

- Blue text represents the characteristics of the input to Enable AI Query.
- Green text represents the characteristics of the preprocessed data.
- Red text represents the output of the model.
- Mauve text represents the time, memory, and storage that is used (resource consumption) by Enable AI Query.

As shown in Figure 6-1, when the size of the AI object is increased by adding more rows, the size of the textual format of AI object increases at the same rate. All columns in the AI object were selected for Enable AI Query processing.

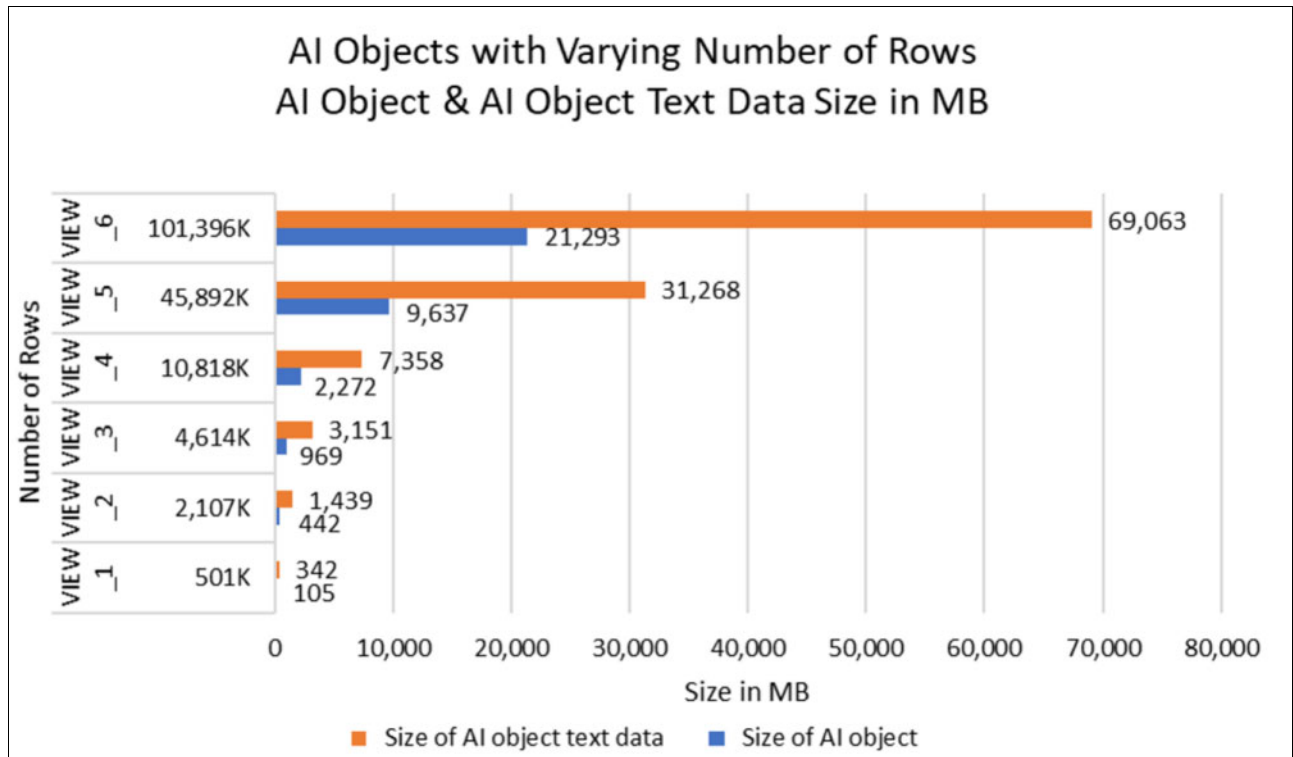


Figure 6-1 The size of AI object textual format data increasing with the size of the AI object

The time to enable AI query and the CPU usage by SQL DI and Db2 also increased at approximately the same rate as the number of rows, as shown in Figure 6-2 on page 217. The CPU usage by SQL DI with eight training threads is almost all zIIP eligible.



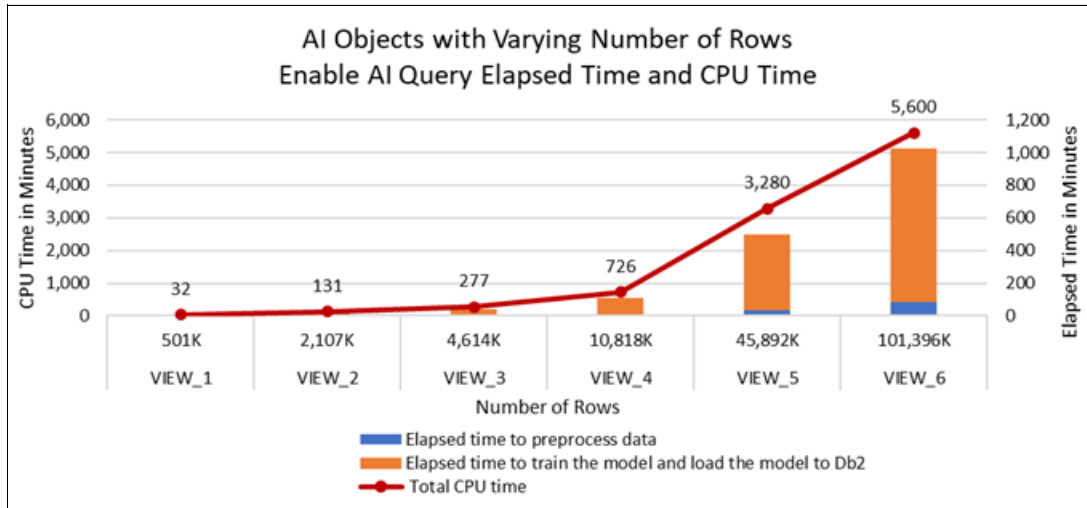


Figure 6-2 Elapsed time and total CPU time breakdown of Enable AI Query processing

The maximum memory that was used and extra DASD storage usage by the zFS file system also increased with the number of rows in the AI object, as shown in Figure 6-3.

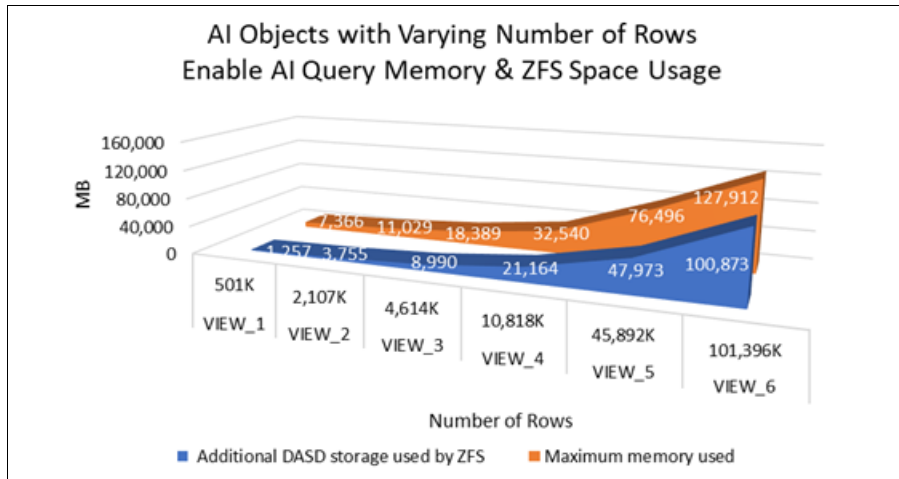


Figure 6-3 zFS space usage and maximum memory usage during Enable AI Query processing

The size of the model increased as the size of vocabulary expanded with the increase in the number of rows, as shown in Figure 6-4. The size of the vocabulary did not increase by much when the number of rows increased from 10 million rows (VIEW\_4) to 45 million rows (VIEW\_5). The reason that the size did not increase much is because the unique values in the additional rows in VIEW\_5 are mostly a repeat of the unique values in VIEW\_4.

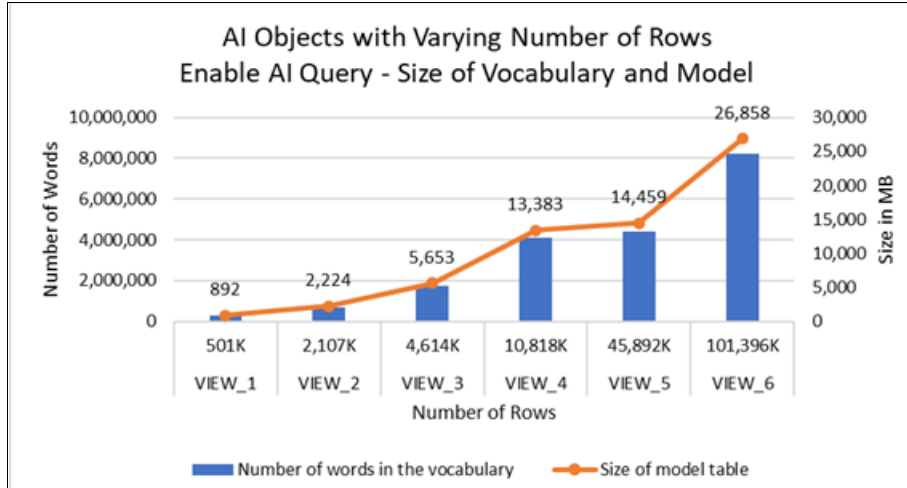


Figure 6-4 The size of the model versus the size of the vocabulary

### Scenario 2

To analyze the influence of the number of columns in determining the resource requirements for training, the second set of measurements ran for Enable AI Query processing on Db2 views with the same number of rows (500 K), varying the number of columns 10 - 60. This measurement demonstrates that as you add more columns for training, the increase in the number of unique values (vocabulary) of the model might not increase at the same rate. The increase in vocabulary size is data-dependent. If the additional columns have only a couple of unique values (for example, YES or NO), the increase in vocabulary size is minuscule. The columns that were added in this set of measurements to increase the size of the Db2 view do not have many unique values.

These measurements are shown in Table 6-2.

Table 6-2 Resource consumption of Enable AI Query with a different number of columns

Enable AI Query	VIEW_1	VIEW_2	VIEW_3	VIEW_4
Number of rows in the AI object <sup>a</sup>	500,734	500,734	500,734	500,734
Size of the AI object (MB)	18.53	34.55	109.66	181.77
Number of numeric or categorical columns that are selected in the AI object to enable AI query	5 / 5	10 / 10	15 / 15	32 / 28
Size of AI object text data (MB)	107.90	191.53	335.49	692.97
Number of words in the training text file (AI object text data) <sup>b</sup>	5,508,074	10,515,414	15,522,754	30,544,774
Number of words in the vocabulary	272,296	272,387	273,530	273,592
Elapsed time to pre-process data (minutes)	0.55	0.73	0.90	1.42

Enable AI Query	VIEW_1	VIEW_2	VIEW_3	VIEW_4
Size of ZLOAD binary size (MB) <sup>c</sup>	895.03	895.33	899.08	899.28
Size of the model table (extra DASD storage that is used by Db2) (MB)	892.04	892.33	896.07	896.27
Elapsed time to train the model and load the model to Db2 (minutes)	0.85	2.53	5.23	9.23
Total elapsed time <sup>d</sup>	1.40	3.27	6.13	10.65
All Db2 address spaces CPU service time (seconds)	2.943	3.501	4.654	7.389
Db2 zIIP eligible (%)	0.412	0.438	0.481	0.528
SQL DI WLM CPU service time (minutes)	6.98	20.31	41.54	73.46
SQL DI zIIP eligible (%)	98.5%	99.4%	99.6%	99.8%
Maximum memory that is used by Enable AI Query (MB)	6,435	6,748	7,480	12,059
Extra DASD storage that is used by zFS (MB)	1,008	1,107	1,264	1,650

- a. Blue text represents the characteristics of the input to Enable AI Query.
- b. Green text represents the characteristics of the preprocessed data.
- c. Red text represents the output of the model.
- d. Mauve text represents the time, memory, and storage that is used (resource consumption) by Enable AI Query.

As we increased the size of the AI object by increasing the number of columns, the size of the textual format of AI object data also increased at the same rate, as shown in Figure 6-5. During our tests, we used different views with a different number of columns, and we selected all columns in the AI object for Enable AI Query.

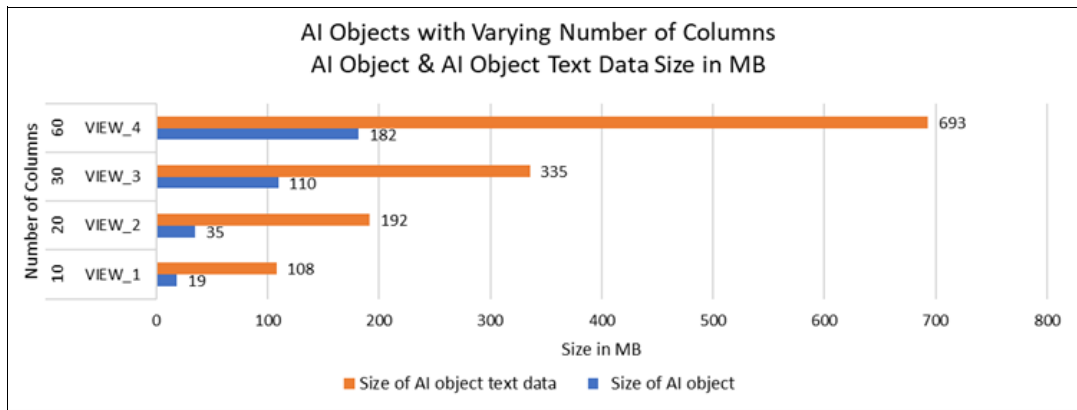


Figure 6-5 Results showing the size of AI object textual format data increasing with the size of the AI object

The elapsed time to perform Enable AI Query processing and the CPU time that is used by SQL DI also increased at the same rate as the number of columns, as shown in Figure 6-6. The CPU time that is used by SQL DI with eight training threads is almost all zIIP eligible.

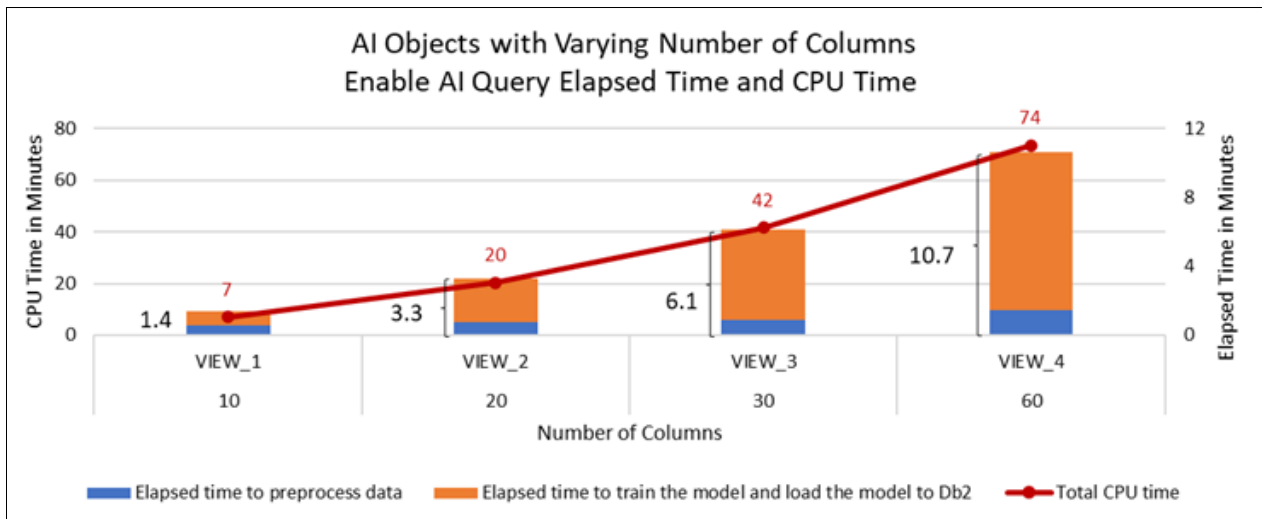


Figure 6-6 Elapsed time and the total CPU time breakdown of Enable AI Query processing

Maximum memory usage and zFS storage that is used by SQL DI increased only gradually as more columns were added to the AI object, as shown in Figure 6-7. This low rate of increase is because the size of the vocabulary did not increase at the same rate as the number of columns, as shown in Figure 6-8 on page 221.

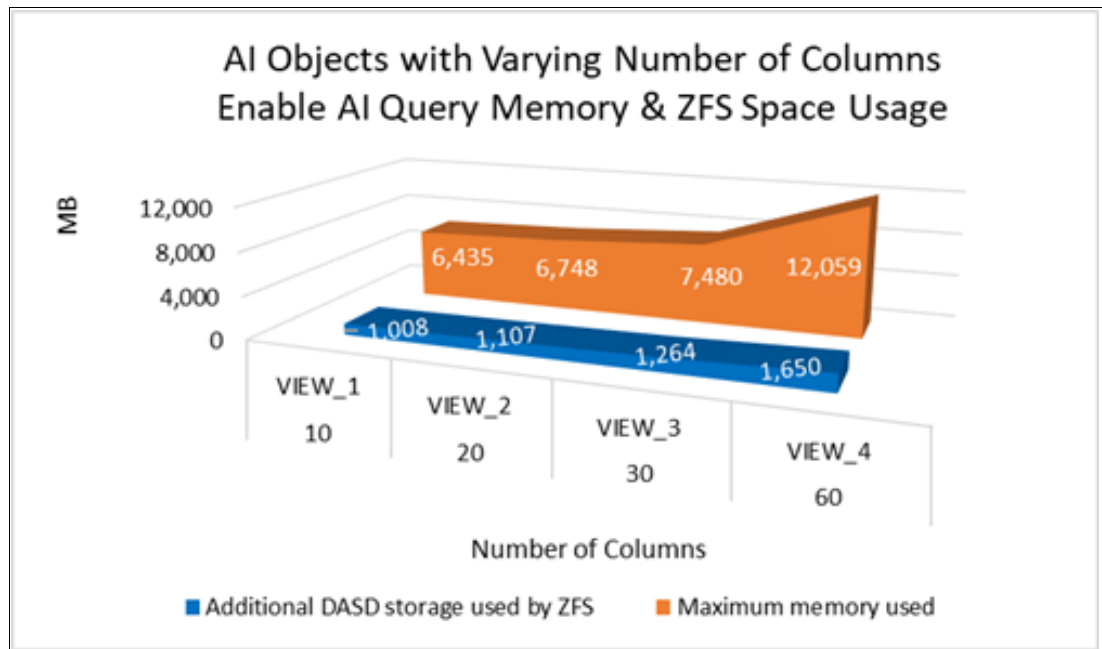


Figure 6-7 zFS space usage and maximum memory usage during Enable AI Query processing

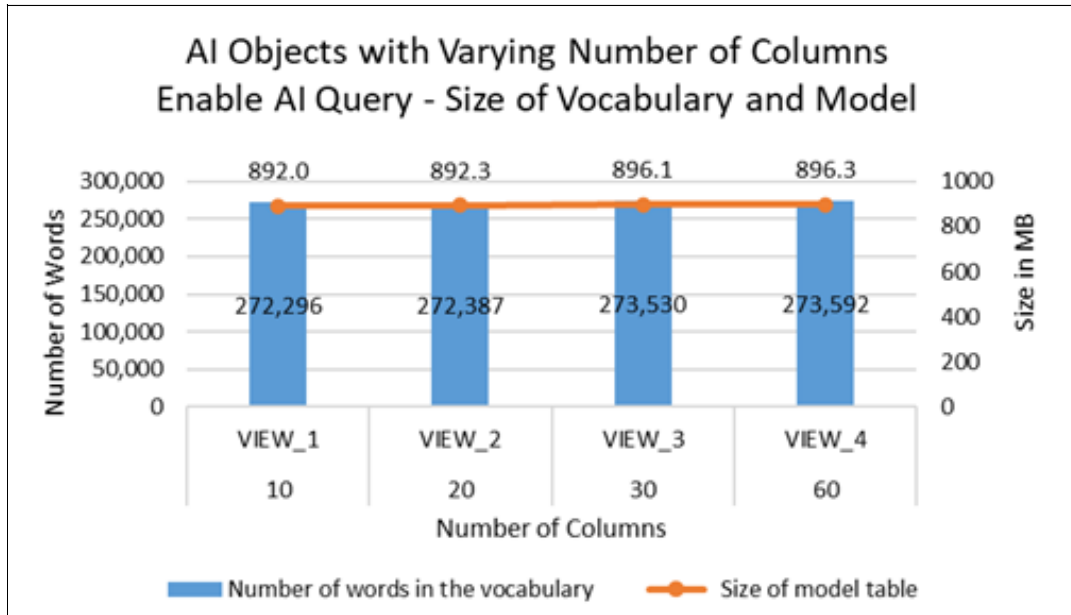


Figure 6-8 Results that show that the size of the model depends on the size of vocabulary

Because the size of the vocabulary (unique values) grew only by a few words as we increased the number of columns, the change to the size of the model table is minimal. The size of the vocabulary for this set of AI objects increased at a slower pace as more columns were added because the added columns have few unique values.

### Scenario 3

A third set of measurements ran to enable AI query on different sizes of Db2 AI objects with a varying number of columns and rows. The first AI object is a VIEW with 29 columns and 500 K rows. The second AI object is a different table with eight columns and 1.3 million rows. This measurement demonstrates that the number of columns or the rows of the input data (AI object) for training cannot solely be used to determine the system resources that are needed for Enable AI Query processing. All columns in VIEW\_1 and TABLE\_1 were selected for model training.

Based on the results in Table 6-3, you cannot plan for all resource requirements based only on the number of rows or number of columns in an AI object. VIEW\_1 has less than half the number of rows compared to TABLE\_1 but consumed more memory and storage resources. VIEW\_1 has 29 columns and TABLE\_1 has eight columns. Having more columns in VIEW\_1 generated a larger input AI object text file for model training because in an AI object text file, every column value is tagged with its column name. A larger vocabulary (unique values) resulted in a much larger model for VIEW\_1. The number of unique values determines the size of the vocabulary in model training and the size of the model or vector table that is stored in Db2. However, because the number of rows in the AI object TABLE\_1 is greater than the number of rows in VIEW\_1, the total elapsed time and CPU time that was needed to enable AI query on TABLE\_1 is more than the total elapsed time and CPU time that was needed to enable AI query on VIEW\_1.

Table 6-3 Characteristics of different AI objects and the resource consumption with Enable AI Query

Enable AI Query	View_1	Table_1
Number of rows in the AI object <sup>a</sup>	500,734	1,371,968
Size of the AI object (MB) - Uncompressed	105.15	236

Enable AI Query	View_1	Table_1
Number of numeric or categorical columns that is selected in the AI object for Enable AI Query	14 / 15	1 / 7
Size of AI object text data (MB) - Uncompressed <sup>b</sup>	341.67	266.64
Number of words in the training text file (AI object text data)	15,022,020	12,347,712
Number of words in the vocabulary	272,271	80,194
Elapsed time to pre-process data (minutes)	0.92	0.83
Size of ZLOAD binary size (MB) <sup>c</sup>	894.95	263.59
Size of the model table (extra DASD storage that is used by Db2) (MB)	891.95	262.71
Elapsed time to train the model and load the model to Db2 (minutes)	3.97	4.65
Total elapsed time (minutes) <sup>d</sup>	4.88	5.48
All Db2 address spaces CPU service time (seconds)	4.997	1.710
Db2 zIIP eligible (%)	49.1%	43.2%
SQL DI WLM CPU service time (minutes)	31.70	37.60
SQL DI zIIP eligible (%)	99.5%	99.7%
Maximum memory that is used by Enable AI Query (GB)	7.19	4.74
Extra DASD storage used by zFS (GB)	1.23	0.53

- a. Blue text represents the characteristics of the input to Enable AI Query.
- b. Green text represents the characteristics of the preprocessed data.
- c. Red text represents the output of the model.
- d. Mauve text represents the time, memory, and storage that is used (resource consumption) by Enable AI Query.

#### **Scenario 4**

The fourth set of measurements were taken to understand the concurrency impact. First, AI Query was enabled on Table\_1, and then AI Query was enabled on two identical tables of the same size and columns concurrently. These two identical tables have the same characteristics as the table that was used in the first measurement, as shown in Table 6-4 on page 223.

Table 6-4 Characteristics of the two AI objects that are used for Enable AI Query concurrency testing

AI query objects	Table_1	Table_2
Number of rows in the AI object <sup>a</sup>	1,371,968	1,371,968
Size of the AI object (MB)	236	236
Number of numeric and categorical columns that are selected in the AI object for Enable AI Query	1 / 7	1 / 7
Size of AI object text data (MB)	266.64	266.64
Number of words in the training text file (AI object text data) <sup>b</sup>	12,347,712	12,347,712
Number of words in the vocabulary	80,194	80,194
Size of ZLOAD binary size (MB) <sup>c</sup>	263.59	263.59
Size of the model table (extra DASD storage that is used by Db2) (MB)	262.71	262.71

a. Blue text represents the characteristics of the input to Enable AI Query.

b. Green text represents the characteristics of the preprocessed data.

c. Red text represents the output of the model.

The elapsed time and resource consumption for Enable AI Query processing for a single and two identical tables concurrently are shown in Table 6-5.

Table 6-5 Elapsed time and resource consumption running two Enable AI Query processes concurrently

Enable AI query	Single enable AI query	Two concurrent enable AI query processes
Total elapsed time <sup>a</sup>	5.52	8.62
All Db2 address spaces CPU service time (seconds)	2.177	3.902
Db2 zIIP eligible (%)	0.456	0.492
SQL DI WLM CPU service time (minutes)	37.82	67.58
SQL DI zIIP eligible (%)	99.7%	99.7%
Maximum memory that is used by Enable AI Query (MB)	5,669	9,377
Extra DASD storage used by zFS (MB)	540	1,080

a. Mauve text represents the time, memory, and storage that is used (resource consumption) by Enable AI Query.

When we enable AI query of two AI objects of the same size concurrently, both Enable AI Query processes completed in approximately 8.62 minutes compared to running a single enable AI query, which took 5.52 minutes. This difference is due to CPU contention during training when running Enable AI Query concurrently. Although each Enable AI Query took 8.62 minutes to complete when running them concurrently, it took 22% less time than when enabling them sequentially (11.04 minutes). This difference is due to little to no CPU contention for certain overlapping phases when Enable AI Query was run for both tables concurrently. These measurements are shown in Figure 6-9.

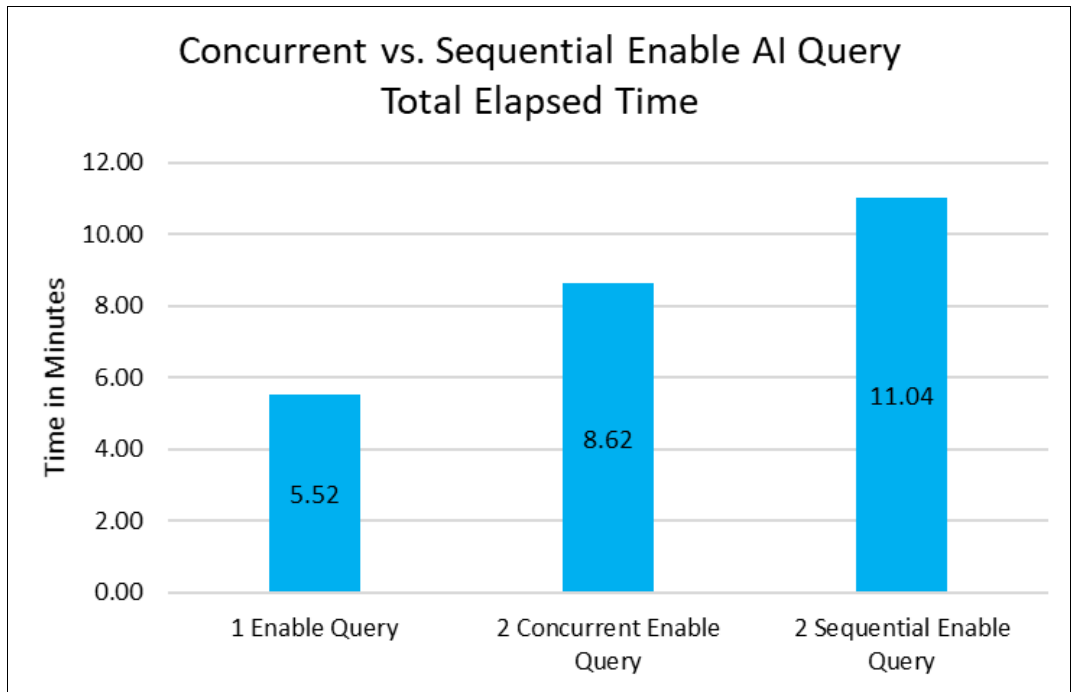


Figure 6-9 The elapsed time of concurrent versus sequential Enable AI Query

### Summary of results

The results of our testing demonstrated that as we keep the number of columns constant and increase the number of rows for an AI object, the size of the textual data grows at the same rate when all columns are selected to enable AI query. The time to train the model and the CPU usage by SQL DI and Db2 also increased at approximately the same rate. The CPU usage by SQL DI is almost all zIIP eligible. Memory usage and extra DASD usage by the UNIX System Services file system also increased with the number of rows in the AI object. The size of the model increased as the size of the vocabulary expanded. In one instance, the size of the vocabulary increased at a slower pace when the number of rows increased from 10 million rows (VIEW\_4) to 45 million rows (VIEW\_5). We observed that this increase is because the unique values in the additional rows in VIEW\_5 are mostly duplicates of the unique values in VIEW\_4.

As we increased the size of the AI object by adding more columns and keeping the number of rows constant, the size of the textual data expanded at the same rate when all columns were selected to enable AI query. The time to train the model and the CPU usage by SQL DI and Db2 also increased at approximately the same rate. The CPU usage by SQL DI is almost all zIIP eligible.



Memory usage and extra DASD usage by UNIX System Services file system increased with the number of columns gradually because the size of the vocabulary did not increase at the same pace as the number of words in the training file. The size of the vocabulary for this set of AI objects did not increase much as more columns were added because the added columns have mostly repeated values. Because the size of the vocabulary (unique values) grew by only a few words as we increased the number of columns, changes to the size of the model table were minimal.

The performance evaluation of different Enable AI Query scenarios demonstrates that as the size of the table or view increases, either by number of columns or rows, the number of words in the textual file increases at the same rate when all columns are selected for Enable AI Query. The time to run the Enable AI Query for AI function and the CPU usage increases at the same rate as the AI object text data size increases. Extra memory usage and DASD storage usage by zFS are a function of both the AI object text data size and vocabulary size. The size of the model and extra storage that is needed by Db2 is directly correlated to the size of vocabulary.

### 6.1.3 Monitoring information

Consider setting up an SQL DI workload for CPU monitoring by completing the following steps:

1. Define a WLM service class for the SQL DI application and WLM service classes for Spark services that are used by SQL DI with their own performance goals.
2. Define an SQL DI workload for these service classes.
3. In the OMVS subsystem type WLM classification rules, specify the address space names of the SQL DI application and Spark services as Qualifier Names to associate them with SQLDI and Spark service classes and report classes. You can use different report classes for different Qualifier Names for reporting granularity.

Now, the RMF Workload Activity Report can be used to obtain CPU time and zIIP usage or eligibility of SQL Data Insights work. For more information about setting up WLM, see [Configuring z/OS workload management for Apache Spark](#).

The starting time and ending time of Enable AI Query processing is captured in the SYSAIDB.SYSAITRAININGJOBS table. You can use the following query to extract the start and end time of Enable AI Query processing and calculate the total elapsed time:

```
SELECT * FROM SYSAIDB.SYSAITRAININGJOBS  
ORDER BY MODEL_ID;
```

You can identify the MODEL\_ID of the AI object by querying the SYSAIDB.SYSAIOBJECTS table.

You can monitor the logs that capture the breakdown of the elapsed time, the size of the vocabulary, and memory usage during the model training phase. These logs are in the UNIX System Services Spark driver directory that is associated with the Enable AI Query run (\$SQLDI\_HOME/spark/worker). The following three logs contain useful information:

- ▶ `base10cluster-local-timestamp.log` contains information about the clusters that are generated for each numeric column that is selected.
- ▶ `ibm_data2vec-local-timestamp.log` contains information about the model training.
- ▶ **stdout** contains time completion information of each phase of Enable AI Query. To view this file, run the following command:

```
vi -W filecodeset=utf-8 stdout
```

## 6.1.4 Usage considerations

This section provides important guidelines and best practices for configuring and using SQL DI.

SQL DI is not required to be configured on the same LPAR where Db2 is. The CPU, memory, and DASD storage resource consumption can be high depending on the size of the AI object. Therefore, plan for the SQL DI configuration to be on an LPAR with workloads that are not mission-critical.

Plan for sufficient zIIP capacity because the high CPU usage by SQL DI is zIIP eligible.

SQL DI can consume all the CPU and memory resources that are available to use. Therefore, limit the use of these resources for the SQL DI workload in z/OS WLM.

Define an SQL DI workload in z/OS WLM with the following characteristics:

- ▶ Include a service class setup for the SQL DI application address space.  
The service class for the SQL DI application address space should be defined with a lower priority than the Db2 address spaces.
- ▶ Include a service class for the Spark address spaces that are used by SQL DI.  
The service class for Spark address spaces should be defined with a lower priority than the SQL DI application address space.

Before you decide to start enable AI query processing, complete the following steps:

1. Identify the columns in the AI object that will never be used or columns whose values do not influence business decisions. By omitting these fields and by using fewer columns, less CPU and storage resources are used, and model training finishes faster.
2. Identify any columns that should be selected as a Key type by using the Analyze Data – Data Statistics window of the AI object. If the number of unique values of a column is the same as the number of rows in the AI object, as a best practice, select this column as a Key type when enabling an AI query.
3. Modify the settings in the Settings SQL DI window, as shown in Figure 6-10.

**Settings**

**Spark**  
Allocate system resources for executing Spark jobs

Driver core (0)

Driver memory (GB)

Executor core

Executor memory (GB)

**CPU threads**  
Specify threads for preprocessing data and training models

Base10 cluster threads

Training threads

**Db2 load utility**  
Customize the Db2 LOAD utility control statement for loading trained models

Utility control statement

Cancel Save

Figure 6-10 SQL Data Insights Settings window

- Specify the number of cores and memory to be used by the Spark driver and executor. Spark uses this setting during the pre-processing phase to transform data into textual format. For more information about deciding how to configure memory size and CPU options for Spark, see [Configuring memory and CPU options](#).
- Specify the number of CPU threads to use during base10 clustering in the pre-processing phase and training phase. We do not recommend that the number of threads exceeds the number of active CPU cores.
- Adjust the **LOAD** parameters to accommodate the size of the model. For a rough estimate of the **ZLOAD** input file (model size), use the Analyze data – Data statistics window to add the number of unique values for columns that you select and specify as categorical type and 20 unique values (clusters) per numerical data type that you select. Then, calculate the total sum of all unique values of categorical data types and all clusters of numerical data type columns. Now, you can estimate the size of the model table data by multiplying the total sum of all unique values by 3300. This result is an overestimation of the model size.

Figure 6-11 shows the Analyze data – Data statistics window for an AI object (not all columns are shown).

Connections / AI objects /

## Analyze data

FMACE1.LOANS\_VIEW\_500K\_10COL Last updated: Oct 7, 2022 5:41 PM [↻](#)

Object details **Data statistics** Column influence

Column name	Db2 data type	# of unique values	Most common value	# of most common values
MI_PERT	DECIMAL	12	0	334647
ORIG_CLTV	DECIMAL	134	80	109632
ORIG_DTI_RATIO	DECIMAL	51	45	25314
MSA_CODE	DECIMAL	422	38060	13767
NUM_UNITS	DECIMAL	4	1	487956
LOAN_SEQ_NUM	CHAR	272144	F19Q10000003	3
FIRST_PAYMENT	DATE	7	2019-03-01	220600
FIRST_TIME_HOMEBUYER_FLAG	CHAR	3	9	226719
MATURITY_DATE	DATE	116	2049-02-01	188213
OCCC_STATUS	CHAR	3	P	436603

Figure 6-11 Data statistics tab of the Analyze data window

**Estimated size of ZLOAD input file (model size)**

$(20 * \text{number of numerical data types}) + (\# \text{ of unique values of categorical data types}) * 3300$

**Estimated size of ZLOAD input file (model size)**

$(20 * 5) + 272273 * 3300$ , which is equivalent to  $272373 * 3300$ , 898,830,900, and ~899 MB

The actual **ZLOAD** input file size in this example is 895 MB. So, 899 MB is a rough estimate of the **ZLOAD** input file.

4. Allocate sufficient storage for the zFS file system where the SQL DI instance is.
5. Add sufficient storage to the storage group that is assigned to the DSNAIDB2 database that holds all model tables.

## 6.1.5 Conclusion

Enabling AI query for using AI functions is the first step to use AI functions in SQL queries. Although this step is not run frequently, the system resource consumption by this task can be intensive as the size of the AI object increases.

However, this demand on system resources is less than the alternative option of hiring a team of data scientists to design ML models and constantly move the data. Also, the CPU that is consumed by SQL DI is mostly zIIP eligible and cost-effective.

## 6.2 Running an AI query

After the neural network model (a collection of numeric vectors) for the AI object is stored in Db2, semantic SQL queries can be run against the AI-enabled object. The semantic SQL query uses the vector model through SQL DI built-in functions to obtain semantic relationships between different entities (vectors).

Db2 13 provides the following three built-in scalar functions that can be run against tables or views that are enabled for AI query to gain hidden insight into the data:

**AI\_SIMILARITY**

The **AI\_SIMILARITY** function determines how similar two entities are within a context (table or view) and returns a similarity score. The cosine distance between the two numeric vectors (entities) is calculated to determine similarity (a score closer to 1.0) or dissimilarity (a score closer to -1.0).

**AI\_SEMANTIC\_CLUSTER**

The **AI\_SEMANTIC\_CLUSTER** function can use a cluster of up to three entities as input and returns a score for other entities in the context (table or view) that have affinity to this cluster. A score closer to 1.0 has a stronger affinity to the cluster.

**AI\_ANALOGY**

The **AI\_ANALOGY** function can be used to compare relationships across two pairs of entities. This function infers the relationship between the first pair, and then assesses the relationship of the second pair to other entities within the context (table or view) and returns a score. A higher score indicates a better analogy.

All three SQL DI built-in scalar functions are zIIP eligible by using a similar mechanism that involves using a child task for query CPU parallelism. This parallelism technique is triggered for SQL DI built-in functions regardless of the parallel degree setting.

The numerical computations that are used by the SQL DI built-in functions are seamlessly accelerated by the IBM zDNN library stack that is in z/OS. The zAIO component of the zDNN library stack chooses the optimal IBM zSystems hardware acceleration.

## 6.2.1 Requirements

The requirements that are shown in the following subsections must be in place to run semantic SQL queries against AI-enabled objects.

### Hardware requirements

An IBM z16, IBM z15, IBM z14, IBM z13, or zEnterprise EC12 system.

### Software requirements

- ▶ Db2 13 (5698-DB2 or 5698-DBV) with FL 500 or later
- ▶ SQL Data Insights FMID HDBDD18
- ▶ z/OS 2.5 or 2.4:
  - zDNN:
    - For z/OS 2.5 with APARs OA62901
    - For z/OS 2.4 with APARs OA62849
  - zAIO:
    - For z/OS 2.5 OA62902
    - For z/OS 2.4 OA62886
  - IBM Z AI Data Embedding Library:
    - For z/OS 2.5 OA62903
    - For z/OS 2.4 OA62887
- ▶ IBM OpenBLAS with the following APARs:
  - PH44479 (for both z/OS 2.5 and 2.4)
  - PH45672 (for z/OS 2.5)
  - PH45663 (for z/OS 2.4)

Objects must be enabled for AI query before you can run semantic SQL queries with AI built-in functions.

## 6.2.2 Performance measurement

Semantic SQL AI query performance measurements ran on different types and sizes of data by using all three currently supported SQL DI built-in functions. All the queries that were used to obtain these performance measurements are shown in Appendix B, “Artificial intelligence semantic queries” on page 409.

### Measurement environment

The following environment was used to conduct this semantic SQL AI performance study:

- ▶ Hardware: IBM z16.
- ▶ CPUs: Three CPs and three zIIPs.
- ▶ LPAR memory: 4 TB.

- ▶ z/OS: 2.5.
- ▶ Db2 version: Db2 13 FL 501.
- ▶ Buffer pool size for data table space: 500 MB.
- ▶ Buffer pool size for data index space: 400 MB.
- ▶ Buffer pool size for model table space: 800 MB.
- ▶ Buffer pool size for where the model index is: 200 MB.
- ▶ SET CURRENT DEGREE = '1' was used by all queries.

### **Measurement scenario description**

The following performance measurement scenarios were conducted to evaluate elapsed time and CPU usage. SE CPU time (CPU time running on zIIP engines) was used as the CPU usage metric for all the measurements that used SQL DI built-in functions because most of the CPU was consumed by zIIP processors.

- ▶ The first set of measurements were conducted to evaluate the difference in elapsed time and CPU usage of AI queries that used user-defined functions (UDFs) and built-in functions (BiFs) for SQL DI functions.
- ▶ The second set of measurements, which used AI queries of different complexities, were performed to assess CPU usage by SQL DI BiFs and zIIP eligibility.
- ▶ The third set of measurements ran to evaluate elapsed time and zIIP CPU usage of the same AI queries running concurrently.
- ▶ The final set of measurements ran to evaluate the scalability of AI queries in terms of elapsed time and zIIP CPU usage as we expand the size of the view by adding more rows.

### **Results**

This section describes the measurement results for the four semantic SQL query scenarios.

#### ***Scenario 1***

Before creating the BiFs in Db2 13, we experimented with UDF SQL DI functions to understand the performance benefit of using BiFs instead of UDFs. The table that was used in the semantic query had eight columns with a cardinality of 1.3 million rows, and its trained model table had 80 K vectors (vocabulary size). All AI semantic SQL queries that were used in this set of measurements are provided in Appendix B, “Artificial intelligence semantic queries” on page 409.

Comparing BiFs to UDFs for SQL DI functions, as shown in Figure 6-12 on page 231, we see that the elapsed time reduction with BiFs is 70% - 89% for the queries that were tested in this experiment.

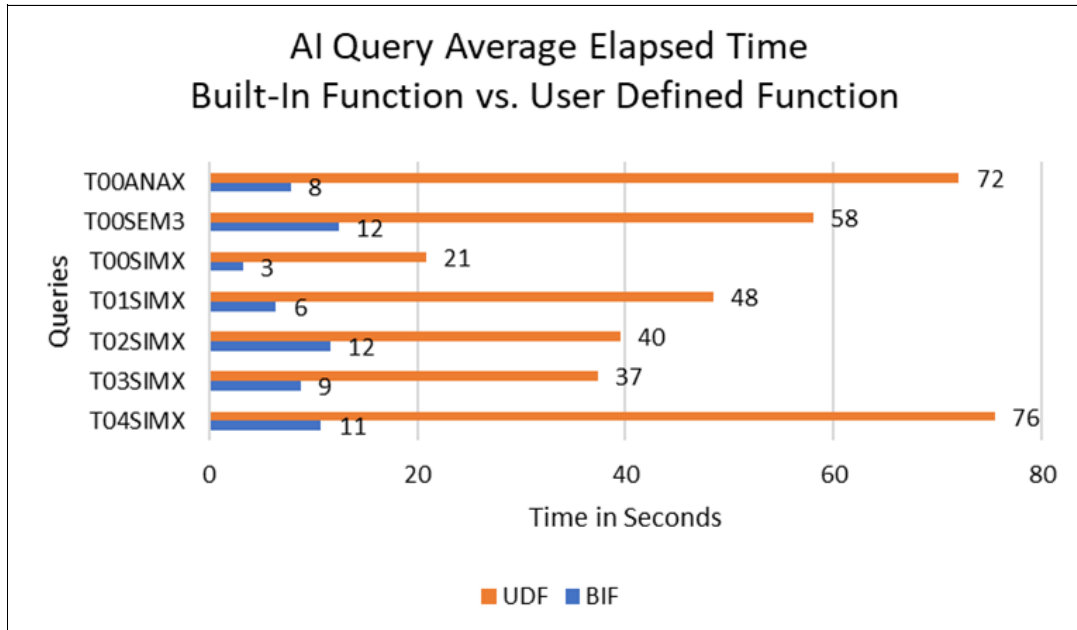


Figure 6-12 Average elapsed time when using BiFs and UDFs for SQL DI functions

Comparing BiFs to UDFs for SQL DI functions shows that the CPU usage reduction with BiFs is 58% - 86% for the queries that were tested in this experiment, as shown in Figure 6-13.

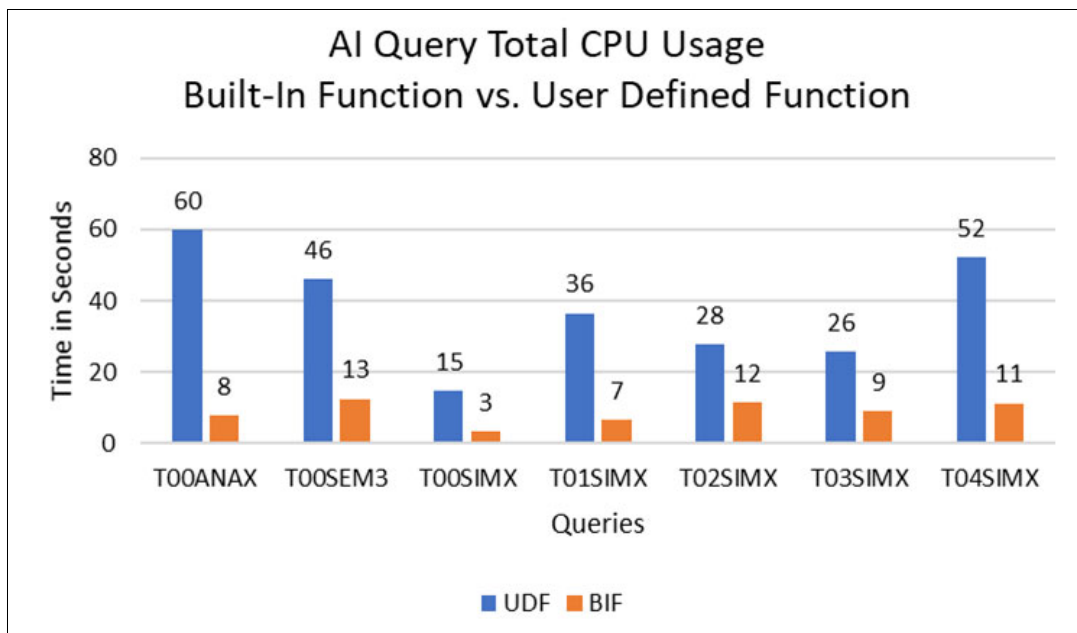


Figure 6-13 Average CPU time when using BiFs and UDFs for SQL DI functions

The performance evaluation of the queries that were tested demonstrate that the elapsed time per query by using BiFs can be reduced by up to 89% compared to using UDFs, and the total class 1 CPU usage can be reduced by up to 86%. Based on these results, we decided to deliver the functions as BiFs in the Db2 engine.

By using BiFs, Db2 can invoke zAIO in the zDNN library stack to trigger IBM zSystems hardware optimization based on processors. Db2 also made the BiF function zIIP eligible, which makes it cost-effective.

**Scenario 2**

AI queries with different complexities ran by using all three supported Db2 SQL DI built-in functions on the same table to demonstrate zIIP usage when invoking SQL DI functions. The table that was used by the semantic queries has eight columns with a cardinality of 1.3 million rows, and its trained model table has 80 K vectors (size of the vocabulary). All AI semantic SQL queries that were used in this set of measurements are shown in Appendix B, “Artificial intelligence semantic queries” on page 409.

The performance results of the different queries that ran by using the **AI\_SIMILARITY** function show that zIIP utilization can be up to 98% based on the number of SQL DI function invocations in the query and the complexity of the queries that ran. Query T00SIMX has only one invocation of the SQL DI function in the query, which means that there is less zIIP eligibility for this query.

The details of the **AI\_SIMILARITY** test results are shown in Figure 6-14.

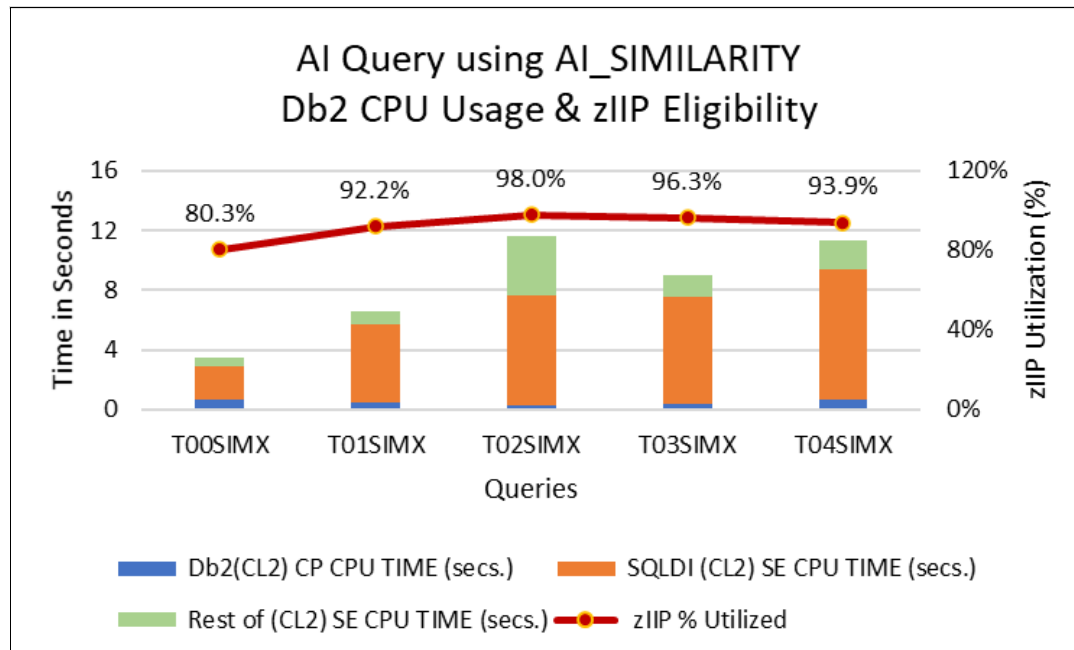


Figure 6-14 Using AI\_SIMILARITY: CPU usage and zIIP utilization

The performance results of the different queries that ran by using the **A\_SEMANTIC\_CLUSTER** function (Figure 6-15 on page 233) show that zIIP utilization can be up to 98% based on the number of entities that is specified for the cluster in the input of SQL DI function invocation. The SQL DI CPU usage increases with the number of entities that is specified in the input cluster.

Query T00SEM1 used one entity in the input cluster, T00SEM2 used two entities in the input cluster, and T00SEM3 used three entities in the input cluster of the SQL DI function.



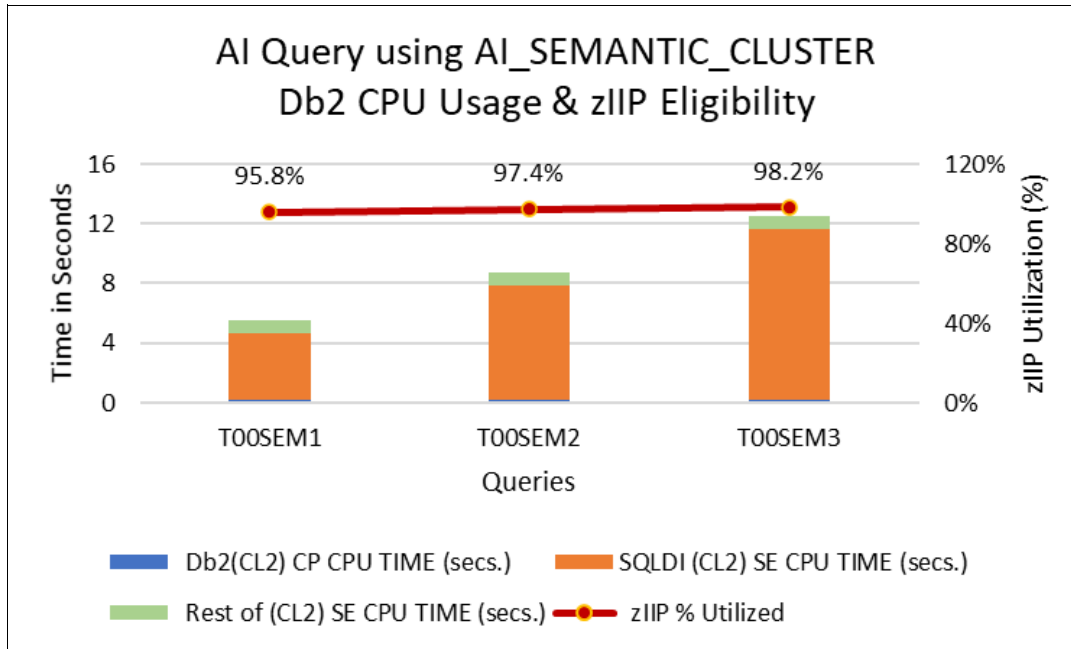


Figure 6-15 Using AI\_SEMANTIC\_CLUSTER: CPU usage and zIIP utilization

The performance results of the two different queries that ran by using the **AI\_ANALOGY** function show that zIIP utilization can be up to 97% based on the number of SQL DI function invocations by the query, as shown in Figure 6-16.

Query T01ANAX has only one invocation of the SQL DI function, which means that there is less zIIP eligibility for the query.

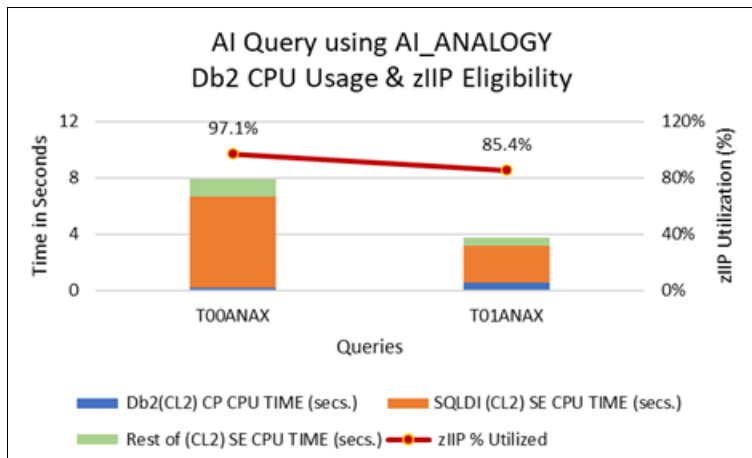


Figure 6-16 Using AI\_ANALOGY: CPU usage and zIIP eligibility

### Scenario 3

This set of measurements ran to evaluate the impact on query-elapsed time and CPU usage as AI semantic queries ran concurrently.

AI semantic queries ran by using all three supported SQL DI built-in functions on the same table with 1.3 million rows, eight columns, and a model size of 80 K vectors or rows. In these measurements, each DDF thread ran the same query against the same table concurrently for the 2-threads and 4-threads tests.

The performance results, which are shown in Figure 6-17, demonstrate that when concurrent AI queries that invoke the `AI_SIMILARITY` function run, the average elapsed time per query increases from waiting for parent and child task synchronization. This increase might be due to contention for CPU. The zIIP usage per query also increases slightly while running concurrent queries. Most of the zIIP CPU usage increase is from running the SQL DI function. CP CPU time is not shown here because it is not a significant part of the total CPU time for this query.

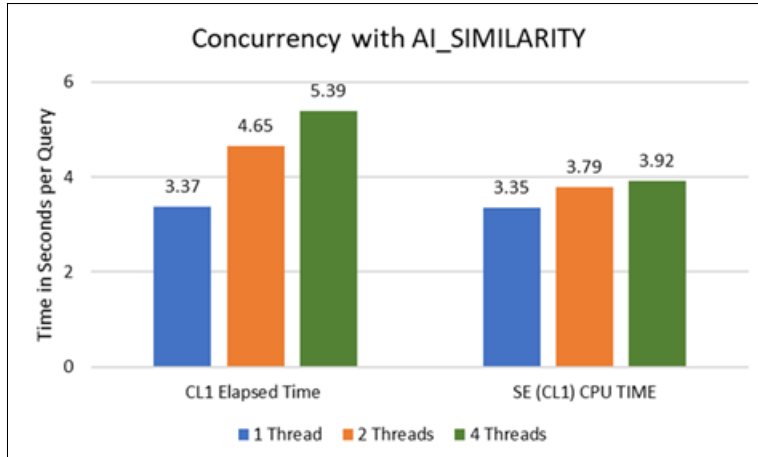


Figure 6-17 Application elapsed time and special engine (zIIP) CPU time per query

The performance results in Figure 6-18 demonstrate that when concurrent AI queries that invoke the `AI_SEMANTIC_CLUSTER` function run, the average elapsed time per query also increases from waiting for parent and child task synchronization. This increase might be due to contention for CPU resources. The zIIP usage per query increases while running concurrent queries. Most of the zIIP CPU usage increase is from running the SQL DI function. CP CPU time is not shown here because it is not a significant part of the total CPU for this query.

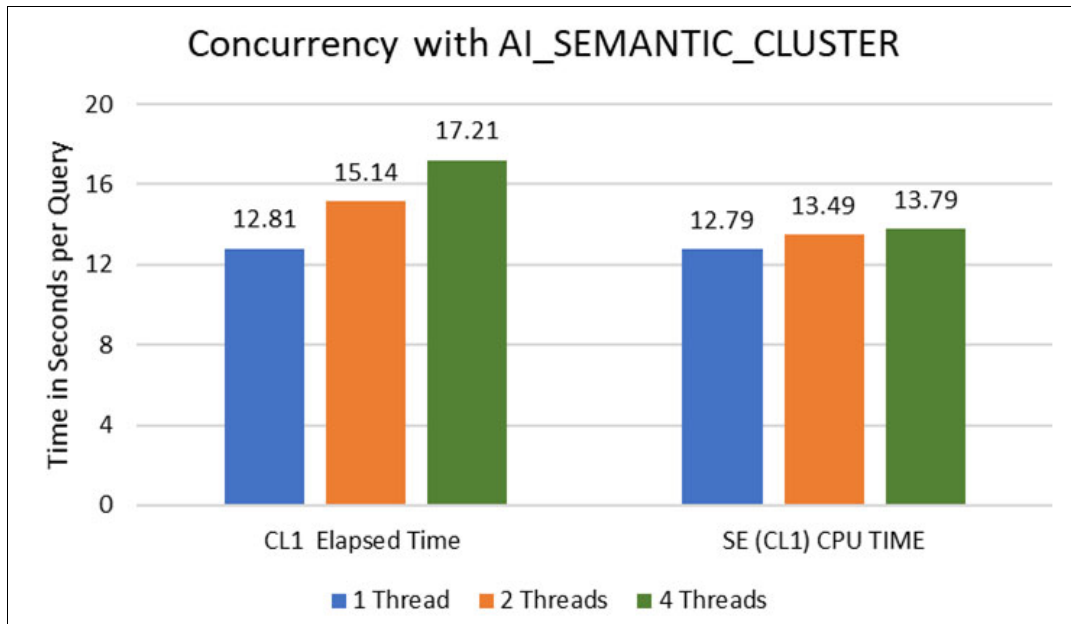


Figure 6-18 Application elapsed time and special engine (zIIP) CPU time per query

The performance results in Figure 6-19 demonstrate that when concurrent AI queries that invoke the `AI_ANALOGY` function run, the average elapsed time per query also increases from waiting for parent and child task synchronization. This increase might be due to contention for CPU resources. The zIIP usage per query also increases while running concurrent queries. Most of the zIIP CPU usage increase is from running the SQL DI function. CP CPU time is not shown here because it is not a significant part of the total CPU for this query.

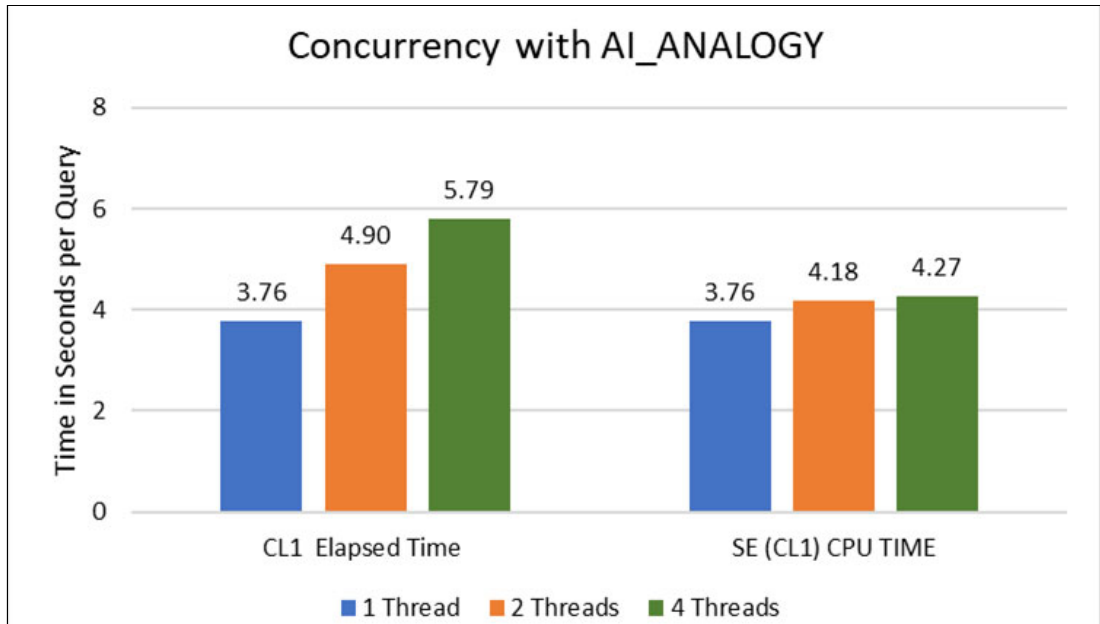


Figure 6-19 Application elapsed time and special engine (zIIP) CPU time per query

### Scenario 4

The last set of performance measurements analyzes the application-elapsed time and CPU usage as we increase the size of the AI object by adding more rows. AI semantic queries ran by using all three supported SQL DI built-in functions on six different views with the number of rows varying from 500 K - 101 million. The evaluation of the model training on these views is shown in “Scenario 1” on page 214. The AI semantic queries that are used here are provided in Appendix B, “Artificial intelligence semantic queries” on page 409.

The performance results in Figure 6-20 show that the elapsed time and zIIP utilization of AI queries that use the `AI_SIMILARITY` function scales upwards when the number of rows in the view increases. For larger views (45 million and 101 million rows), the join method that was used was a merge scan join as opposed to a nested loop join, which was used for querying smaller size views.

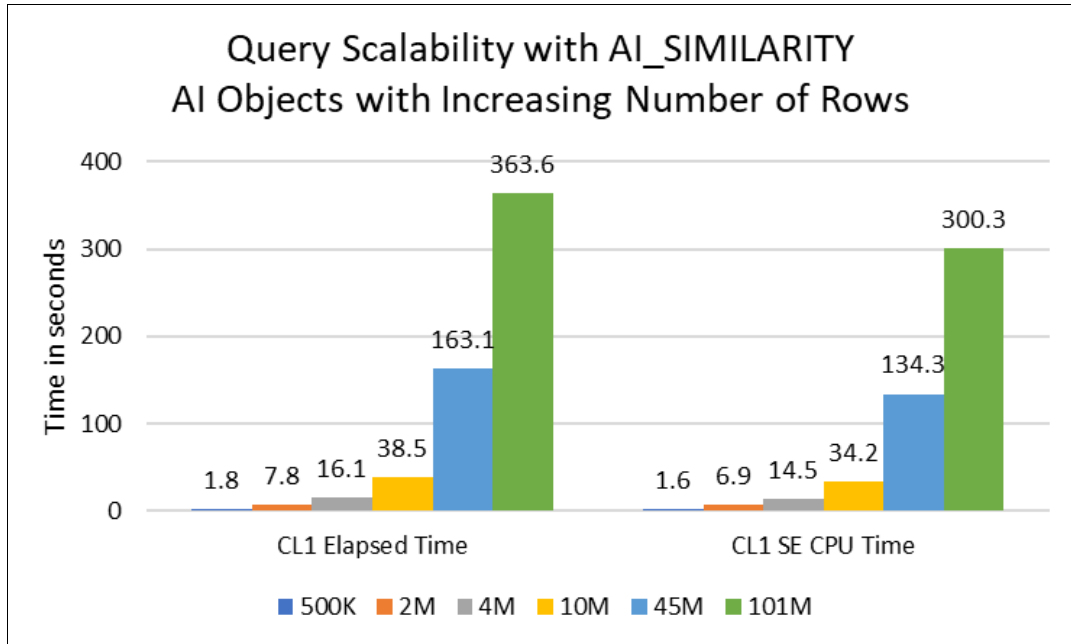


Figure 6-20 Application elapsed time and special engine (zIIP) CPU time per query

The performance results in Figure 6-21 on page 237 show that the elapsed time and zIIP utilization of AI queries that use the `AI_SEMANTIC_CLUSTER` function scale upwards when the number of rows increases until 10 million rows. For 45 million rows, the elapsed time and CPU usage did not increase because the optimizer chose a different access path by using a different index for one of the base tables for the view with 45 million rows, which yielded reduced elapsed time and CPU usage.

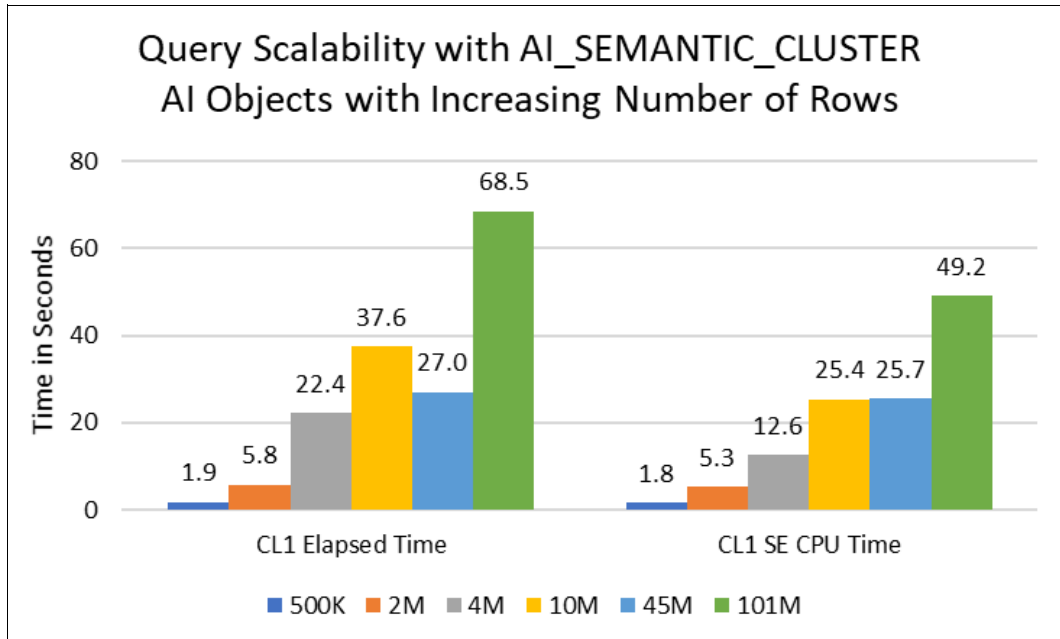


Figure 6-21 Application elapsed time and special engine (zIIP) CPU time per query

The performance results in Figure 6-22 show that the elapsed time and zIIP utilization of an AI query with the **AI\_ANALOGY** function scales upwards with an increase in the number of rows.

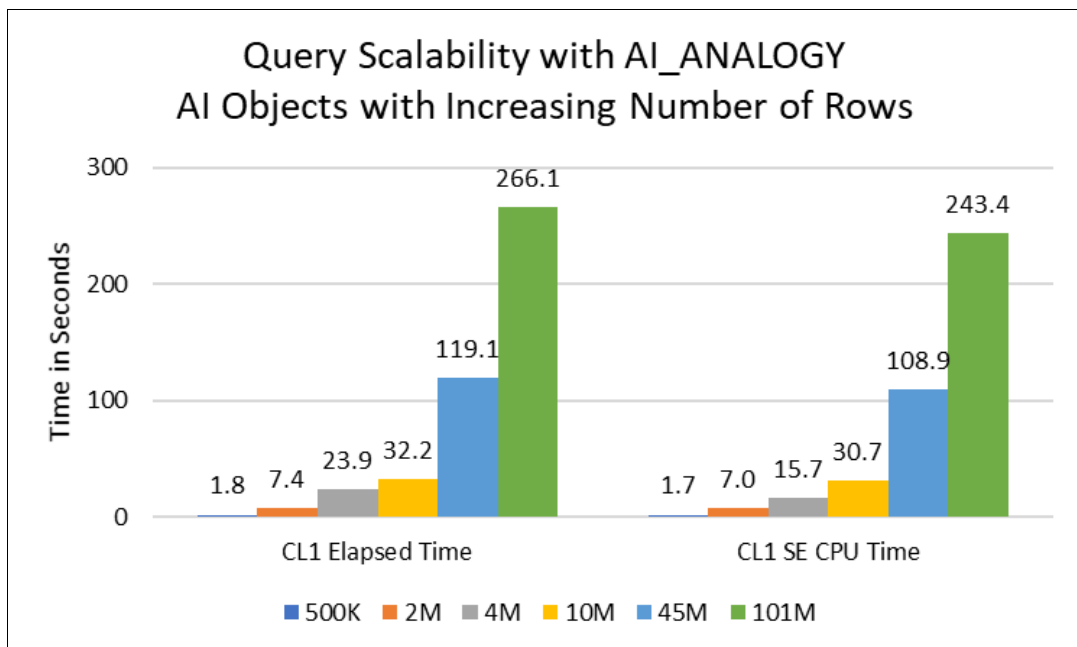


Figure 6-22 Application elapsed time and special engine (zIIP) CPU time per query

### Summary of results

The performance evaluation of semantic AI queries that use both UDFs and BiFs for the SQL DI functions indicated that SQL DI functions should be implemented as BiFs. By using a built-in function, Db2 can invoke IBM zSystems hardware optimization for processors.

All measurements of different AI queries with SQL DI built-in functions demonstrate that the SQL DI function is zIIP eligible. There was no general-purpose central processor (GPC) time that was attributed to running SQL DI functions for any of these measurements.

Concurrently running AI queries takes longer to finish than running a single query. The increase in elapsed time is due to waiting for parent and child task synchronization. This increase might be due to contention for CPU resources. The zIIP usage per query also increases slightly while running queries concurrently. Most of the increase in zIIP CPU usage is from running SQL DI functions.

The elapsed time and CPU usage scale upwards as we increase the number of rows of the AI object for the queries and test cases that are used here. One exception is the AI query when using the **AI\_SEMANTIC\_CLUSTER** function on the view with 45 million rows. In this case, the Db2 optimizer used a different access path by using a different index on one of the base tables, which yielded better elapsed time and zIIP CPU usage. This result might be because the view that was created with 45 million rows is on all the rows in the base tables.

## 6.2.3 Monitoring information

A new section for monitoring SQL DI usage (QWAC\_AIDB\_STATS) is available in the Instrumentation Facility Component Identifier (IFCID) 3 accounting trace. New fields in this section monitor the number of invocations of SQL DI functions, elapsed time to process SQL DI, and CPU usage by both general-purpose central processors (GPU) and IBM specialty engines (zIIP). All these times are part of the class 2 (in-Db2) time.

<b>QWAC_AIDB_FNS_ELAP</b>	Elapsed time that is spent processing SQL DI functions.
<b>QWAC_AIDB_FNS_CP</b>	GPC time that is spent processing SQL DI functions.
<b>QWAC_AIDB_FNS_ZIIP</b>	IBM specialty engine (zIIP) time that is spent processing SQL DI functions.
<b>QWAC_AIDB_COUNT</b>	The number of entry and exit events that are performed by SQL DI functions.

## 6.2.4 Usage considerations

Db2 objects must be enabled for AI query before you can run AI semantic queries by using SQL DI functions. To specify a column as an argument in an SQL DI function, it must be a column that was selected to enable AI query.

You can use SQL DI functions in SQL statements in the **SELECT** clause or as part of the **WHERE** clause.

The syntax and restrictions for writing queries by using the new SQL DI scalar functions can be found in the following topics:

- ▶ [AI\\_SIMILARITY](#)
- ▶ [AI\\_SEMANTIC\\_CLUSTER](#)
- ▶ [AI\\_ANALOGY](#)

SQL DI functions may run on zIIPs, so they are cost-effective. However, plan for zIIP capacity to accommodate the CPU resources that are needed for complex queries with multiple invocations of SQL DI functions or for running multiple AI queries concurrently.

## 6.2.5 Conclusion

The CPU usage by SQL DI function is mostly zIIP eligible. SQL DI built-in functions interface with zAIO for numerical computations, which trigger the most optimal hardware acceleration.

With SQL DI built-in functions, you can run AI semantic SQL queries directly on tables or views that use the learned information in the vector model to gain insights into the data. You can make faster business decisions without offloading the data from Db2 and running queries in a data warehouse.

## 6.3 Summary

The SQL Data Insights feature, which is available in Db2 13, brings AI capability directly to the Db2 engine without the need to hire experienced data scientists for designing ML models of Db2 data.

Enabling AI query on any Db2 table or view triggers the building of a neural network model of the data in Db2 object and storing the resulting vector model in the same Db2 subsystem. We demonstrated that the elapsed time to enable AI query and the resource consumption in terms of CPU, memory, and disk storage usage can increase as the size of the input data for model training expands. We also showed that the performance to enable AI query might depend on both the number of columns that is selected for enabling AI query and the cardinality of the input data. The size of the vector model that is stored in Db2 is data-dependent. The number of unique values in the input data directly correlates to the size of the model.

Although the system resource consumption by the model training task can be intensive as the size of the AI object increases, this step is not frequently run, and the CPU that is used during this phase is mostly zIIP eligible and cost effective.

Semantic AI queries that operate on any Db2 data on which you enabled AI query provide a deeper understanding of the data to make effective business decisions faster, which can be achieved in a cost-effective manner because the SQL DI functions are eligible to run on zIIPs. Additionally, because the data can remain on the Db2 subsystem where it is, the overhead that is required to move that data is eliminated. All SQL DI built-in functions use zAIO to choose the most optimal hardware during semantic AI query execution.







# Application concurrency

Db2 13 introduces several enhancements that affect application concurrency. The following enhancements are described in this chapter:

- ▶ Inserting improvements into partition-by-growth table spaces
- ▶ Index look-aside optimization
- ▶ Index management diagnostic and serviceability enhancements
- ▶ Application timeout and deadlock control
- ▶ Enhanced support for profile tables

## 7.1 Inserting improvements into partition-by-growth table spaces

Balancing space usage and performance is important. It is more important for partition-by-growth (PBG) table spaces when deciding whether to use or reuse space in the current partition, move to another existing partition, or create a partition. Db2 13 enhanced some of the algorithms to improve insert performance and insert success rate for PBG table spaces.

### 7.1.1 Requirements

To leverage these improved algorithms, you must be using Db2 13 at function level (FL) V13R1M100 and later.

### 7.1.2 Retrying a search of previously failed partitions

In Db2 12, when an application cannot obtain a conditional lock on a partition from PBG table space, it continues and tries the next partition. When all partitions are checked and no available space is found to perform the insert, the application might receive an SQLCODE -904 with a reason code 00C9009C or 00C90090. These errors do not provide adequate information for diagnosing the cause of the problem.

Figure 7-1 illustrates this problem. The target partition, partition 3, is full. Db2 continues to check partition 4 but cannot get the conditional lock, which is commonly caused by a utility concurrently running on this PBG table space or another application exclusively locking the partition. When partition 5 is tried, the same thing occurs. Db2 goes back to search partition 1, and then partition 2, which are both full. Now that Db2 checked all partitions (and did not manage to insert the row), the **INSERT** statement fails with SQLCODE -904 with reason code 00C9009C, which does not give enough information to understand the reason for the failure. A new partition is not created because the insert operation cannot access partitions 4 and 5 because it does not have adequate space information to determine whether these partitions are full. This behavior avoids adding extra partitions when existing partitions are not fully used.

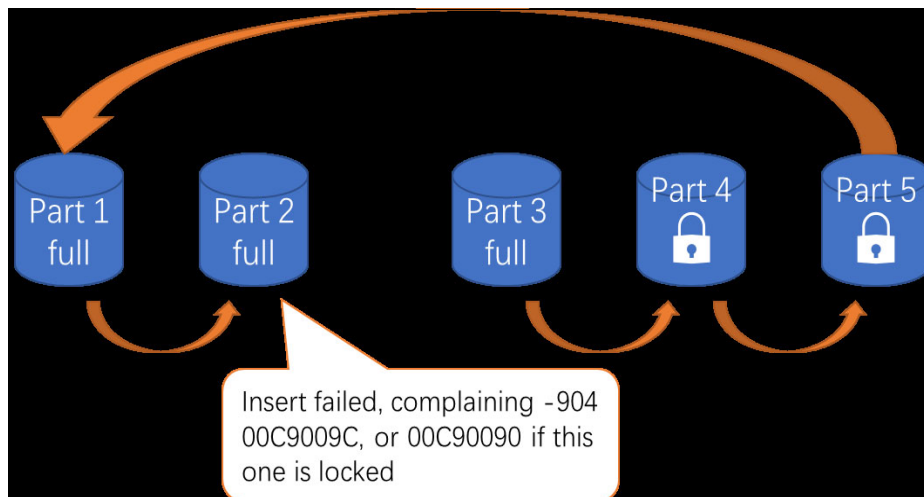


Figure 7-1 Insert failed with -904 with 00C9009C or 00C90090

Db2 13 improved the insert algorithm so that it remembers the partitions that it previously failed to obtain a conditional partition lock for (in our example, partitions 4 and 5) and retries them in turn after the first search round. If Db2 still cannot obtain the partition lock, an SQLCODE -911 or -913 with a reason code 00C90088 is returned, which provides details about the lock holder and waiter. Figure 7-2 and Figure 7-3 show sample error information that you might see if Db2 cannot obtain the required partition lock.

```

----- SQLCA -----
Error code: -913
SQLERRMC: 00C90088;00000210;FALCDB01.TSFALC03.00000022
  token 0: 00C90088
  token 1: 00000210
  token 2: FALCDB01.TSFALC03.00000022
SQLERRP: DSNXRINS
SQLERRD(1): -110
SQLERRD(2): 13172746
SQLERRD(3): 13172872
SQLERRD(4): -1
SQLERRD(5): 0
SQLERRD(6): 0
SQLWARN1:
SQLWARN2:
SQLWARN3:
SQLWARN4:
SQLWARN5:
SQLWARN6:
SQLWARN7:
SQLWARN8:
SQLWARN9:
SQLWARNA:
SQLSTATE: 57033

```

Figure 7-2 Sample -913 SQLCODE and associated error information

```

DSNT375I  +DD2H PLAN=DISTSERV WITH 965
          CORRELATION-ID=Insert_ORDER
          CONNECTION-ID=SERVER
          LUW-ID=G91E81DB.G6E9.DB11496FE546=17

THREAD-INFO=USRT001:9.30.129.219:USRT001:Insert_ORDER_HIST:DYNAMIC:26
:*:<::9.30.129.219.1769.DB11496FE546>
          IS DEADLOCKED WITH PLAN=LCKPART1 WITH
          CORRELATION-ID=RUNLKPT1
          CONNECTION-ID=BATCH
          LUW-ID=NATIVE.DSNDD1H.DB114964ABA4=133
THREAD-INFO=USRT001:BATCH:USRT001:RUNLKPT1:DYNAMIC:1:*:*
          ON MEMBER DD1H
DSNT501I  +DD2H DSNILMCL RESOURCE UNAVAILABLE 966
          CORRELATION-ID=Insert_ORDER
          CONNECTION-ID=SERVER
          LUW-ID=G91E81DB.G6E9.DB11496FE546=17
          REASON 00C90088
          TYPE 00000210
          NAME FALCDB01.TSFALC03.00000022

```

Figure 7-3 Example of DSNT375I error message on the console

Compared to Db2 12, the Db2 13 retry logic can increase the success rate of highly concurrent insert applications by using PBG table spaces. A higher success rate means that more rows are inserted successfully with less effort of intervention.

## Performance measurements

To evaluate this enhancement, a performance study was conducted by using a PBG table space that is defined with and without the **MEMBER CLUSTER** clause.

### Scenario description

To test the new algorithm, we prepared a PBG table space that is defined with **MAXPARTITIONS 120**, and with 23 partitions filled with data (**DSSIZE 1G**). Each partition has a small amount of space for new inserts. The test run inserts 200,000 rows by using 10 concurrent threads on each member of a two-way data-sharing group (one row per commit). Concurrent jobs are used to mimic other applications that are holding locks on partitions randomly. The PBG table space is defined with page-level locking, **APPEND NO**, and it uses insert algorithm 1.

The test completes the following steps:

1. Loads 60 million rows of data (unloaded from a random insert workload) to fill up 20+ partitions.
2. Randomly deletes 60,000 rows to create a small amount of space in each partition.
3. Randomly inserts 200,000 rows by using 10 threads on each member; inserts one record per commit with a concurrent job to randomly lock about 10 partitions; and holds the lock for 0.5 seconds before locking the next batch.
  - Configuration 1: Two-way data sharing, one PBG, one index, page-level locking with **MEMBER CLUSTER**, insert algorithm 1, **APPEND NO**, **DSSIZE 1G**, **NUMPARTS** not specified, **MAXPARTITIONS 120**, **PCTFREE 0**, **FREEPAGE 0**
  - Configuration 2: Same as Configuration 1, except without **MEMBER CLUSTER**

### Measurement environment

The following system configuration was used to make these measurements:

- ▶ IBM z14 hardware with two Coupling Facilities (CFs) at CF LEVEL 23, each with three dedicated CPUs
- ▶ Logical partition (LPAR) CPU: Eight general CPs
- ▶ LPAR memory: 48 GB (for each of the two LPARs that are used in this testing)
- ▶ z/OS 2.4
- ▶ Db2: Two-way data-sharing group on two LPARs, one running Db2 12 at FL 509 and one running Db2 13 at FL 500

### PBG table space with MEMBER CLUSTER

Figure 7-4 on page 245 shows the total rows that were inserted successfully and the number of unsuccessful inserts when using a PBG table space with the **MEMBER CLUSTER** option.

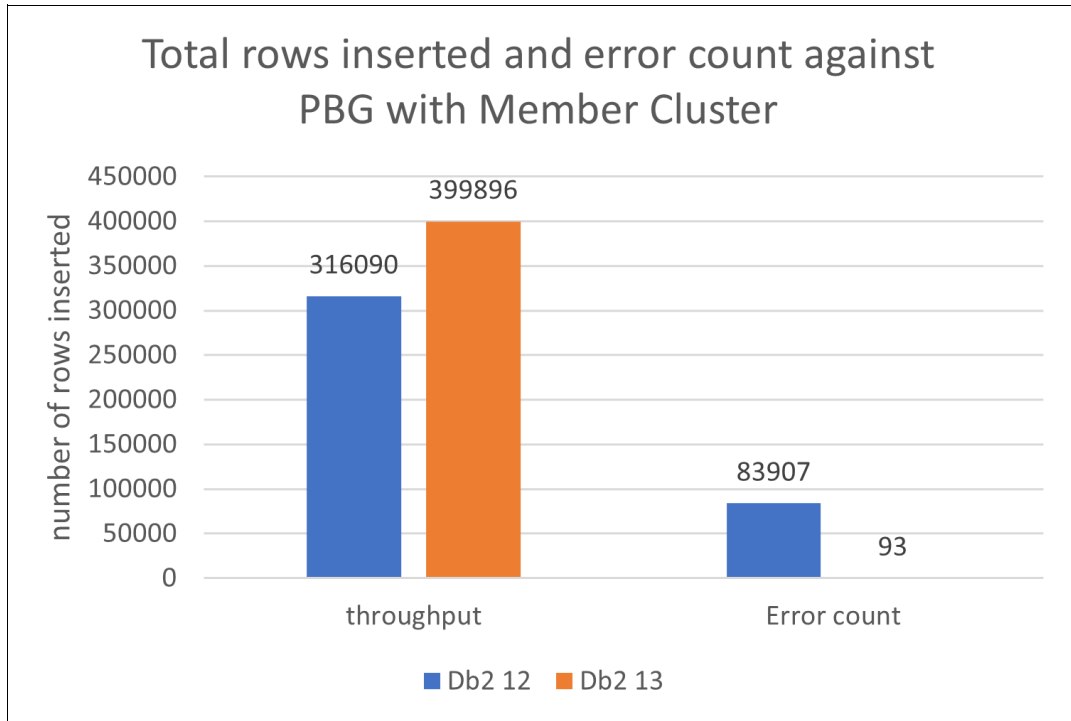


Figure 7-4 Total rows that are inserted and error count against a PBG table space with MEMBER CLUSTER

With the retry logic, Db2 13 can insert 26.5% more rows into the table, and the number of errors is reduced by 99.9%. The 93 errors are all SQLCODE -913 instead of -904, and each of them provides detailed lock holder and victim information that you can use to identify the application that is causing the locking problems.

No obvious extra space is required when the new algorithm is used. Db2 13 tends to insert more rows into the last partition or newly grown partition.

Figure 7-5 shows the average Db2 class 2 CPU per commit and average class 2 elapsed time per commit for the test scenario against a PBG table space with MEMBER CLUSTER. Compared to Db2 12, Db2 13 provides an improvement in CPU time. The CPU reduction comes from a reduced amount of getpage activity because for a MEMBER CLUSTER table space, the new algorithm keeps a retry list for a prepared INSERT statement. When the next row is inserted, rather than searching every part, Db2 tries the partitions in the retry list first. In Db2 12, the high getpage activity is due to the concurrent lock part jobs, which cause Db2 to keep searching for a part that is not locked and has space for an insert.

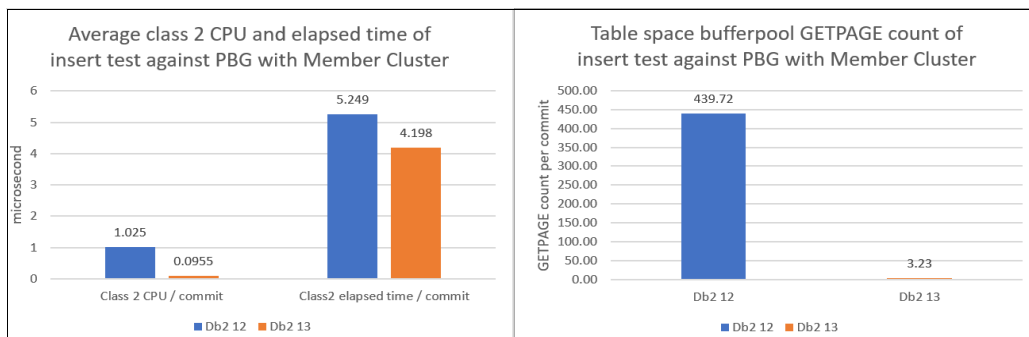


Figure 7-5 Class 2 CPU, elapsed time, and getpage against PBG with MEMBER CLUSTER

**PBG table space without MEMBER CLUSTER**

Like “PBG table space with MEMBER CLUSTER” on page 244, when Db2 13 uses a PBG table space without a **MEMBER CLUSTER**, the number of errors also are reduced by 99.9%. 3.2% more rows can be inserted into the table, as shown in Figure 7-6.

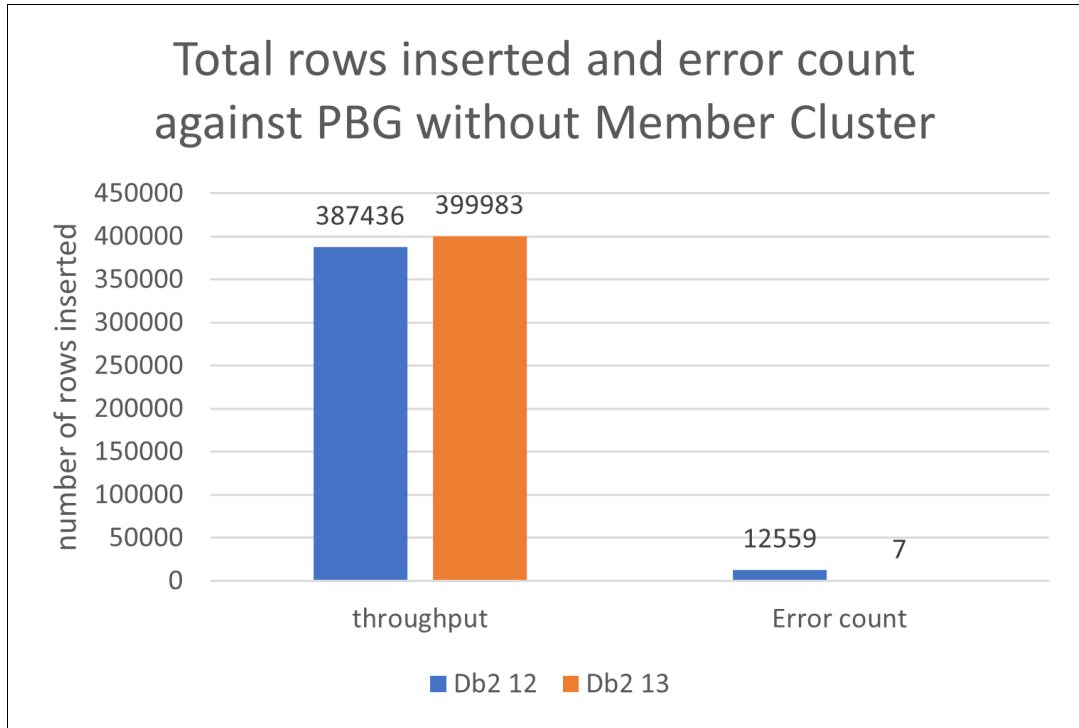


Figure 7-6 Total rows that are inserted and error count against the PBG table space without MEMBER CLUSTER

Testing revealed an 18% savings in class 2 CPU time for the PBG without **MEMBER CLUSTER** scenario, although some increase in elapsed time because of longer global contention time was observed, as shown in Figure 7-7. With the Db2 13 retry logic, Db2 can wait longer to get a partition lock for some rows. A slight elapsed time increase is acceptable considering the drastic reduction in the number of insert failures because those inserts typically drive rollbacks and retries in applications, which are typically much more disruptive.



Figure 7-7 Class 2 CPU, elapsed time, and getpage against PBG without MEMBER CLUSTER

### 7.1.3 Smarter cross-partition search

Inserting data into a PBG table space with limited free space can cause a cross-partition search, which triggers the checking of other partitions in ascending or descending order. When partitions are physically allocated but not all of them are used, or when trailing empty partitions are created by **REORG** after it reclaims deleted space, a peculiar insert behavior can occur that results in many empty partitions in the middle of the table space.

Figure 7-8 shows how such a situation can result from using descending partition search. Because the search is performed in ascending or descending order randomly, empty partitions can be present in the middle of the table space when the descending search is used. For example, suppose that the candidate page is in partition p7, but there is no available space. Db2 (randomly) performs a descending search, and because partitions p6 - p1 are full, Db2 wraps around to partition p18. This partition is empty and has free space, so Db2 inserts the row in p18. Now, the last partition p18 is in use, but previous partitions p13 - p17 are empty.

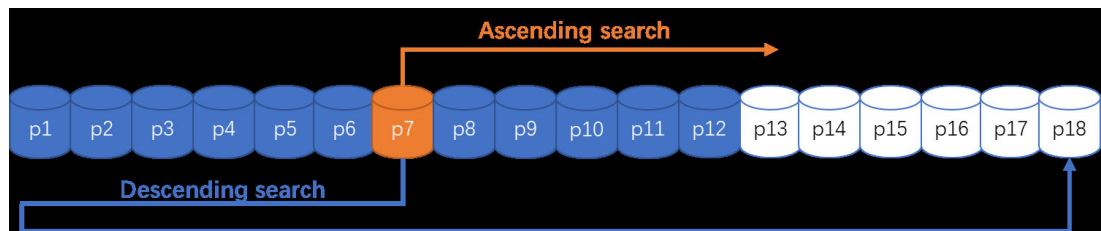


Figure 7-8 Ascending and descending search during PBG cross-partition space search

To avoid this situation, Db2 13 enhances the descending search algorithm so that when a descending search reaches partition 1 and there is no space, instead of going directly to the last physical partition, Db2 determines the next best target partition to search by referring to real-time statistics (RTS) and by caching the usage information. This process is shown in Figure 7-9. When the descending partition search reaches p1, it switches to p12 (if partition 12 has enough space).

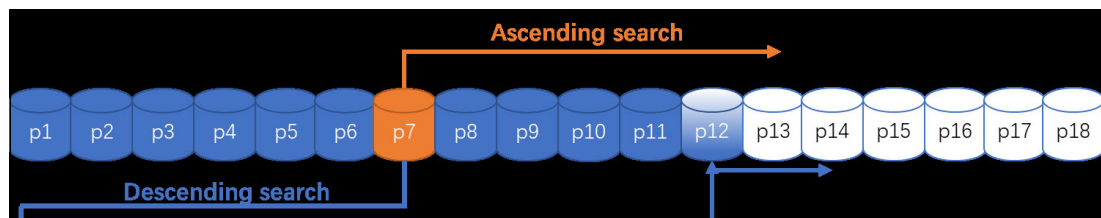


Figure 7-9 Behavior of the new descending search algorithm

The internal tracking information is kept in memory and updated when an insert operation traverses through existing partitions since the last physical open of the page set. Therefore, each data-sharing member can have different tracking information based on the workload activity on each member. When a page set is newly opened, insert performance gradually improves until Db2 caches the last non-empty partition. With this new algorithm, even if there are trailing empty partitions, Db2 does not go to the last physical partition, so there are no empty partitions in the middle of the table space.

#### Performance measurements

To evaluate this enhancement, a performance study was conducted by using a PBG table space that was defined with and without the **MEMBER CLUSTER** keyword.

### ***Scenario description***

To test the new algorithm, a PBG table space was prepared with 100 partitions pre-allocated with a data set size of 1 GB (**DSSIZE 1G**). Seventeen of the partitions were fully filled with data.

Then, the test inserted 2 million rows by using 50 concurrent threads on each member of a 2-way data-sharing group. Concurrently, there were 50 concurrent delete threads, which deleted 2 million existing rows on each member. The table space was defined with page-level locking, **APPEND NO**, and insert algorithm 1. Testing was done by using both a **MEMBER CLUSTER** PBG table space and a non-**MEMBER CLUSTER** PBG table space. The initial target page (based on the clustering index) for the insert workload was mainly in partition 1.

The test completes the following steps:

1. Randomly inserts 100,000 rows with high key values (40,000,001 ~ 40,100,000) into an empty PBG table to establish clustering that triggers an initial search from partition 1.
2. Loads the PGB with 40 million rows (1 ~ 40,000,000) initially. After that, 17 or 18 partitions have data when **COMPRESS NO** is used, and five partitions have data when **COMPRESS YES** is used.
3. Runs the test to insert randomly 2 million rows by using 50 threads on each member (40,100,001 ~ 44,100,000, with one row per batch and 10 rows per commit).

Concurrently, randomly deletes 2 million rows by using 50 threads on each member (3,000,000 ~ 5,000,000 and 20,000,001 ~ 22,000,000).

Both members of the 2-way data-sharing group are configured identically. The object that is used in both configurations is also identical, except for the **MEMBER CLUSTER** attribute:

- Configuration 1: Two-way data sharing, one PBG, one index, page-level locking, **MEMBER CLUSTER**, insert algorithm 1, **DSSIZE 1G**, **NUMPARTS 100**, **MAXPARTITIONS120**, **PCTFREE 0**, **FREEPAGE 0**, **APPEND NO**
- Configuration 2: Two-way data sharing, one PBG, one index, page-level locking, non-**MEMBER CLUSTER**, insert algorithm 1, **DSSIZE 1G**, **NUMPARTS 100**, **MAXPARTITIONS120**, **PCTFREE 0**, **FREEPAGE 0**, **APPEND NO**

### ***Measurement environment***

The following system configuration is used to make these measurements:

- ▶ IBM z14 hardware with two CFs at CF LEVEL 23, each with two dedicated CPUs
- ▶ LPAR CPU: Eight general CPs
- ▶ LPAR memory: 48 GB (for each of the two LPARs)
- ▶ z/OS 2.4
- ▶ Db2: Two-way data-sharing group on two LPARs, one running Db2 12 at V12R1M509 and one running Db2 13 at V13R1M500

### ***MEMBER CLUSTER PBG table space***

Figure 7-10 on page 249 shows how rows are distributed across partitions after the test scenario runs. With Db2 12, the first 18 partitions contain data, and two partitions at the end (part 99 and part 100) contain data. With Db2 13, these last physical partitions no longer have data; instead, partitions 18 and 19 are filled with more rows with no empty partitions in between.



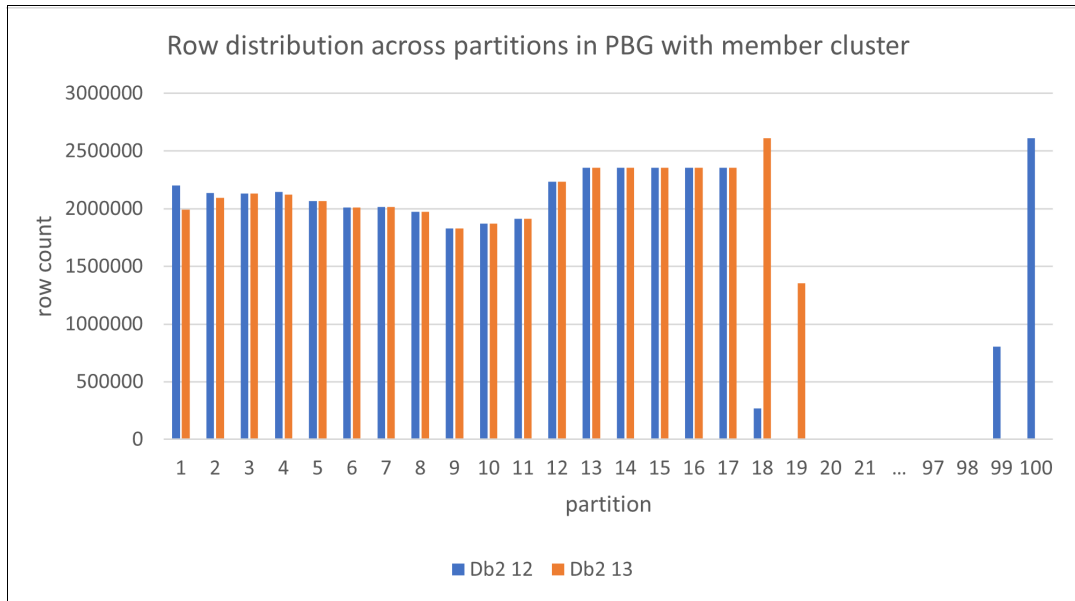


Figure 7-10 Data distribution across partitions for a PBG with MEMBER CLUSTER

Figure 7-11 shows the class 2 CPU and elapsed time per commit of the insert threads and the table space getpage activity of the test run by using the **MEMBER CLUSTER** PBG table space. Note the 14.6% improvement in CPU time, which is mainly because of the reduction in table space getpage activity. Because concurrent delete activity is occurring, the lock retry strategy (see 7.1.2, “Retrying a search of previously failed partitions” on page 242) is triggered, and Db2 retries the parts in the retry list rather than searching each partition every time. The result is a reduction in the number of getpages. The performance of the insert activity can vary depending on the nature of the workload, for example, whether there is concurrent delete activity, how many parts are filled with data, and how full these partitions are. As always, your results might vary.

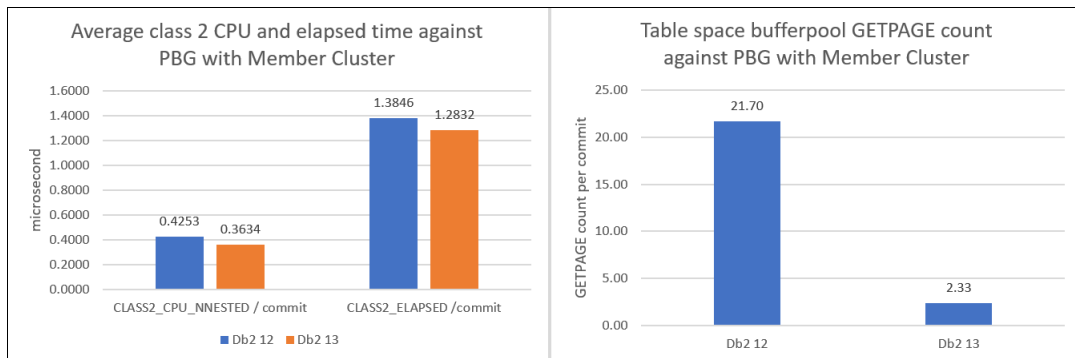


Figure 7-11 Class 2 CPU, elapsed time, and getpage against PBG with MEMBER CLUSTER

### Non-MEMBER CLUSTER PBG table space

This scenario uses the same setup and workload that is described in “Scenario description” on page 248. The only difference is that the PBG table space is not using the **MEMBER CLUSTER** attribute.

Figure 7-12 shows how rows are distributed across partitions after an insert workload completes. As with the **MEMBER CLUSTER** PBG test, in Db2 13 there are no more empty partitions in between.

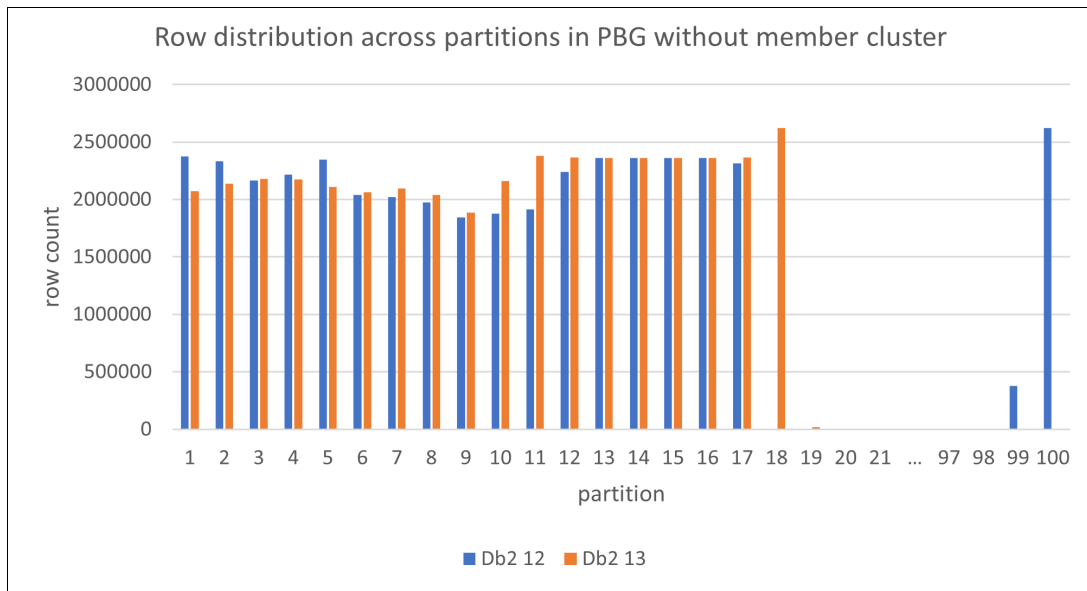


Figure 7-12 Data distribution across partitions for a PBG without **MEMBER CLUSTER**

From a performance perspective, the non-**MEMBER CLUSTER** PBG table space behavior is different than the **MEMBER CLUSTER** case. With the non-**MEMBER CLUSTER** attribute, Db2 must start from the target partition based on the candidate page that is determined by the clustering index (partition 1 in this test) for every row that is inserted, which means that Db2 must go through the entire cross-partition search process. No obvious performance differences were observed when comparing these results against the Db2 12 test results.

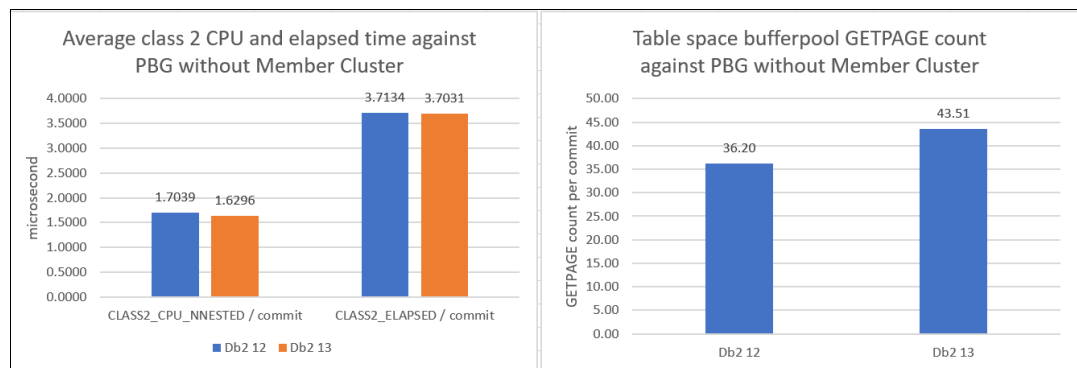


Figure 7-13 Class 2 CPU, elapsed time, and getpage against PBG without **MEMBER CLUSTER**

For this scenario, in which the candidate page (based on the clustering index) is in partition 1, Db2 12 directly goes to partition 100 during a descending search, and it inserts rows in that partition, establishing index clustering there. Db2 13 tries to find the last non-empty partition (partition 17 in this scenario) during a descending search. When the page set is first opened, this last non-empty partition is unknown to Db2. It takes some time for Db2 to learn that the last partition with data is partition 17 (before partition 18 is filled with data) and establish a clustering order there, making partition 17 the target partition (based on the candidate page).

Now, Db2 knows that partition 17 is the last non-empty part and caches it in the memory, which is why there are some extra getpages in the Db2 13 run compared to the Db2 12 run, as shown in Figure 7-13 on page 250.

Further testing showed that when the last non-empty partition (partition 17 in this case) is reached and a cluster order is created there, Db2 13 outperforms Db2 12, with up to a 34% CPU time improvement and a 23% getpage reduction.

### 7.1.4 Conclusion

In a highly concurrent workload environment, Db2 13 can achieve a higher insert success rate against tables in PBG table spaces. With the enhanced retry insert logic, Db2 13 reduces the possibility of failing to get partition locks and reduces the effort to diagnose the errors, with no extra CPU cost, and some elapsed time increase.

The improved descending search algorithm for PBG table spaces eliminates the behavior of inserting into partitions at the end when pre-allocated partitions exist, which results in empty partitions in the middle. Performance is not affected overall, except that when a non-MEMBER CLUSTER table page set is newly opened, it takes some time for Db2 to identify the last non-empty partition to perform the descending search partition efficiently.

Insert performance depends on many factors. The lab tests that are described in this section try to explain how the improved algorithm works. You might observe different results for your workloads, depending on the PBG configuration, space usage, data distribution, insert pattern, concurrent workloads that run together with the insert activity, and so on. The insert behavior and performance at run time can vary.

## 7.2 Index look-aside optimization

Index look-aside is a technique that Db2 uses to improve index access efficiency. Db2 tracks index value ranges and checks whether a requested index entry is on the same leaf page as the previous request. If the key that is being requested is on the same leaf page as the previous request, Db2 returns the entry without traversing the index B-tree or even performing a getpage request for the leaf page. For index-intensive operations, the avoided B-tree traversals and the getpage requests can save CPU processing time.

Index look-aside is always enabled for read operations (select and fetch). For update activities (insert, update, and delete operations) before Db2 13, index accesses for SQL insert and SQL delete operations can use index look-aside for clustering indexes and non-clustering indexes with high cluster ratios based on catalog statistics. Before Db2 13, index look-aside is not supported for update operations.

With Db2 13, index look-aside support is expanded in scope. Index look-aside can be enabled for all indexes during SQL insert, update, and delete operations regardless of the cluster ratio in catalog statistics. Db2 13 monitors each index's look-aside efficiency for insert, update, and delete operations by using look-aside hit counters within a commit scope. With these counters, Db2 turns on look-aside for an index when look-aside can be beneficial and disables look-aside for an index when it fails to meet certain criteria. This dynamic management of look-aside for each index ensures that look-aside is used efficiently without depending on catalog statistics, which can be inaccurate.

Index look-aside support for operations other than insert, update, and delete is unchanged in Db2 13 and behaves the same as in Db2 12.

## 7.2.1 Requirements

The expanded use of index look-aside is enabled by default in Db2 13 starting at FL 100.

## 7.2.2 Performance measurements

This section describes the measurements that were taken to assess the performance of the index look-aside enhancements that are available in Db2 13.

### Measurement environment

The following setup was used for these measurements:

- ▶ IBM z15 hardware
- ▶ Two CFs at CF LEVEL 24, each with three dedicated CPUs
- ▶ LPAR memory: 512 GB (for each of the two LPARs for the data-sharing tests)
- ▶ LPAR CPU: Eight general CPs and two IBM zSystems Integrated Information Processors (zIIPs) (for each of the two LPARS for the data-sharing tests)
- ▶ z/OS 2.4
- ▶ Db2:
  - Base measurements: Db2 12 at FL 510
  - Db2 13 measurements: Db2 13 at FL 100

### Measurement scenario description

To fully evaluate the expanded index look-aside capabilities in Db2 13, we designed a series of performance workloads that considered most of the factors that affect index look-aside usage. The workloads cover insert, update, and delete operations with different data access patterns. We used indexes with varying levels of cluster ratios, and the workloads were run in both non-data-sharing and data-sharing environments. We also evaluated the performance of these workloads with their package bound with and without **RUNSTATS** information, and with and without fast traversal blocks (FTBs) enabled.

The indexes that were used by the workloads and their characteristics are listed in Table 7-1. The base table space that the workloads use is a PBR table space with 32 partitions.

*Table 7-1 Index look-aside workloads: Indexes and their characteristics*

Index	IXPBR01	IXPBR02	IXPBR03	IXPBR04	IXPBR05	IXPBR06
Buffer pool	BP21	BP22	BP23	BP24	BP25	BP26
Cluster ratio %	98	88	68	48	18	0
Clustering index?	Yes	No	No	No	No	No
Unique?	Yes	No	No	No	No	No

The different workloads have the following characteristics:

► Insert workloads

These workloads insert 9,900,000 rows into a table with 100,000 rows with multiple-row inserts (MRI) by using a rowset size of 100, that is, 100 rows are inserted per commit. The insert testing consisted of the following four scenarios:

– Scenario 1

The 100 rows of data for a single MRI operation are arranged in such a way that for each index, the first data rows in the MRI array match the clustering index's order sequentially and the rest of the data rows have random keys. In the first scenario, the package is bound without prior **RUNSTATS**, which means that Db2 does not know each index's cluster ratio except that IXPBR01 is the clustering index.

For example, take index IXPBR02. The key column values for the first 90 rows of data in the MRI array are sequential. The last 10 rows of data have randomly generated column values for the IXPBR02's key. For example, the key order in the MRI rowset array for the IXPBR02 key column values could be:

1,2,3,...89,90, 67862, 84064,47461,12202,3112,69471,435,47896,17100,91237

– Scenario 2

The data insert order is the same as scenario 1. However, the package was bound after **RUNSTATS** ran on the table space, so that, except for knowing that IXPBR01 is the clustering index, Db2 also knows that IXPBR02 has a high cluster ratio.

– Scenario 3

The 100 rows of data for a single MRI operation are arranged in such a way that for each index, the first data rows in the MRI array have randomly generated column values for the index keys, and the rest of the data rows in the array have column values that match the index' clustering order sequentially. In scenario 3, the package was bound without prior **RUNSTATS**, which means that Db2 does not know each index's cluster ratio except that IXPBR01 is the clustering index.

For example, take index IXPBR04. The key column values for the first 50 rows of data in the MRI array are randomly generated. The rest of the 50 rows of data have sequential column values for the IXPBR04 keys. For example, the key order in the MRI array for the IXPBR04 key column values could be:

67862, 84064,47461,12202,3112,69471,435,47896,17100,91237,...,1,2,3...50

– Scenario 4

The data insert order is the same as scenario 3. However, the package was bound after **RUNSTATS** ran on the table space, so that, except knowing IXPBR01 is the clustering index, Db2 also knows that IXPBR02 has a high cluster ratio.

► Update workloads

For the workloads that are used in update testing, all packages were bound without prior **RUNSTATS**. The update testing consisted of the following two scenarios:

– Scenario 1

Updating 10 random rows in a single commit scope

– Scenario 2

Sequentially updating 100 rows of data starting from a random row in a single commit scope

- ▶ Delete workload

Only a single scenario was used to test delete operations. The scenario sequentially deletes 100 rows of data per commit, without **RUNSTATS** information.

## Measurement results

The index look-aside optimization that was introduced in Db2 13 provides solid improvements of the workload performance. The performance results are described in three parts:

- ▶ Insert workloads
- ▶ Update workloads
- ▶ Delete workloads

**Note:** The performance results that were observed by using the workloads are highly dependent on the workload behavior, the cluster ratios of the indexes, data characteristics, testing environments, and so on. The performance results from this Db2 13 enhancement are likely to be different in your environment when running your workloads.

### *Insert workloads performance*

The main way that index look-aside improves the performance of an application is by reducing getpage operations. A getpage reduction was observed for all the insert workloads that were tested, which resulted in CPU savings for all these workloads.

By analyzing the getpage counts of the indexes with different cluster ratios for the insert workloads, the following results were observed:

- ▶ Getpage counts for IXPBR01 are similar between the Db2 12 and Db2 13 tests because index look-aside is enabled for both Db2 12 and Db2 13 for clustering index IXPBR01.
- ▶ Without **RUNSTATS**, although IXPBR02 has a high cluster ratio of 88%, in Db2 12, index look-aside is not enabled for this index. However, in Db2 13, the dynamic monitoring algorithm ensures that index look-aside is enabled for IXPBR02 when suitable, and the number of getpages for IXPBR02 is reduced by close to 80%.
- ▶ When **RUNSTATS** ran and the IXPBR02 high cluster ratio updated in the catalog statistics for the index, index look-aside is enabled for IXPBR02 in Db2 12, and the getpage counts are similar between the Db2 12 and Db2 13 tests.
- ▶ For indexes IXPBR03, IXPBR04, and IXPBR05 with lower cluster ratios, with or without **RUNSTATS**, Db2 12 does not enable index look-aside for them. In Db2 13, with the dynamic enablement of index look-aside, getpages were reduced for all three of these indexes. The higher the cluster ratio, the greater the getpage reduction for that index.
- ▶ For index IXPBR06, which has a cluster ratio of 0, the getpage count of the Db2 12 and Db2 13 tests are almost the same. Because index look-aside for IXPBR06 does not provide any benefits, Db2 13 dynamically turns off look-aside at run time, but Db2 12 never enabled index look-aside for IXPBR06.
- ▶ Running the same test with or without FTBs allocated results in similar percentages of getpage reduction for the same indexes.
- ▶ Running the same test by using data-sharing and non-data-sharing configurations also results in a similar reduction in the number of getpages for the same indexes.

Figure 7-14 and Figure 7-15 depict the total getpage counts of all the indexes that are accessed by the insert workloads. For workload characteristics for scenarios 1, 2, 3, and 4, see “Measurement scenario description ” on page 252.

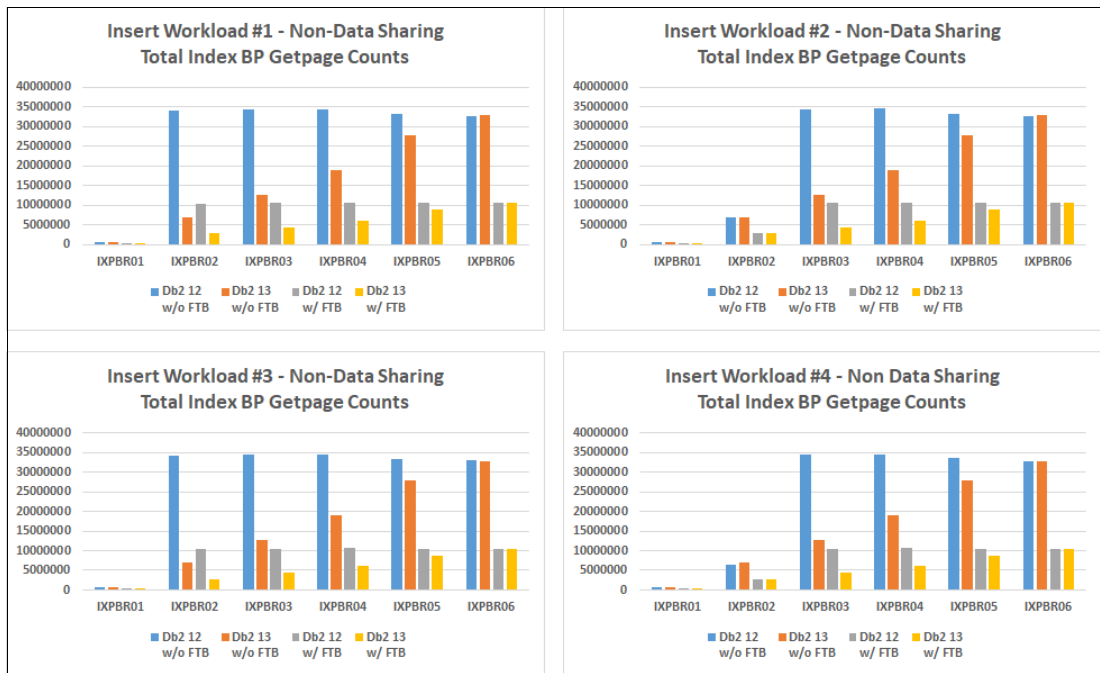


Figure 7-14 Insert workloads: Total index getpages in a non-data-sharing environment

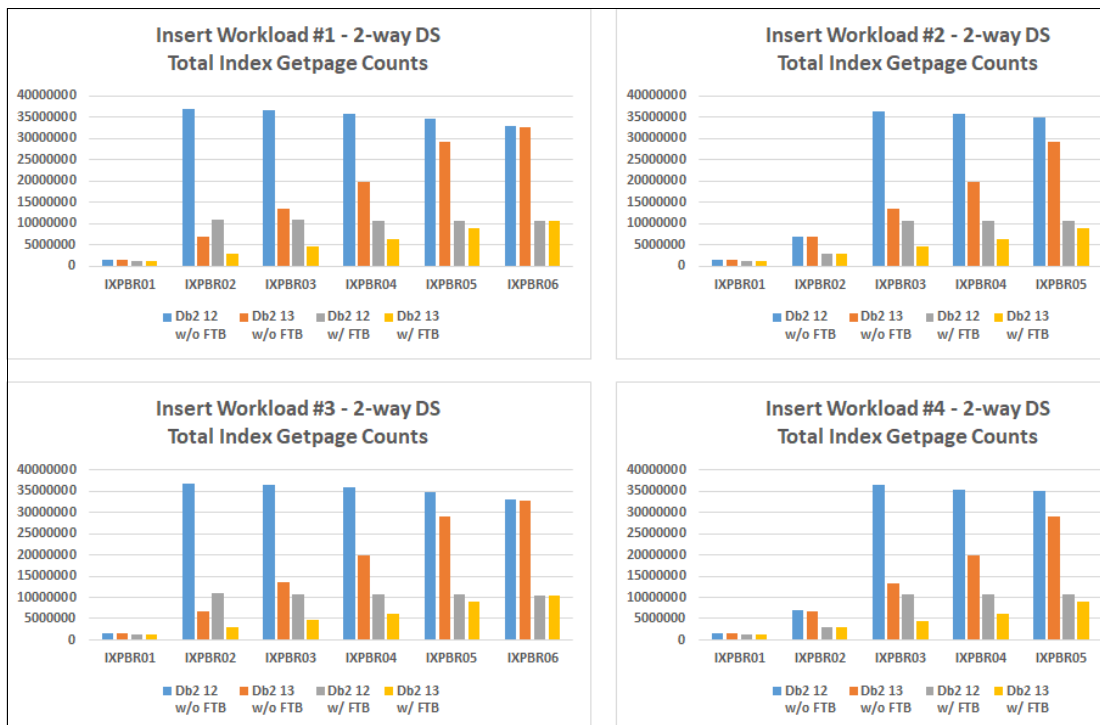


Figure 7-15 Insert workloads: Total index getpages in a two-way data-sharing environment

All the getpage changes that are described in this section indicate that the Db2 13 look-aside enhancements are performing effectively by keeping index look-aside on for an index when it is beneficial and turning off index look-aside when it does not help. With the getpage reductions, we also see CPU time benefits of different degrees depending on whether the workloads are run in data-sharing environment, and whether FTBs are allocated for indexes. Because class 2 CPU time was reduced for most of the tests, we also see a class 2 elapsed time reduction for the Db2 13 tests.

For the non-data-sharing tests, the following results were observed when comparing the Db2 13 performance results to the Db2 12 results:

- For the tests without FTBs allocated, with a total getpage reduction of 30 - 41% for all the indexes (depending on whether **RUNSTATS** ran for the Db2 12 tests), we observed a 5 - 9% total class 2 CPU time reduction. We also see some amount of class 2 elapsed time reduction. For more information, see Figure 7-16.

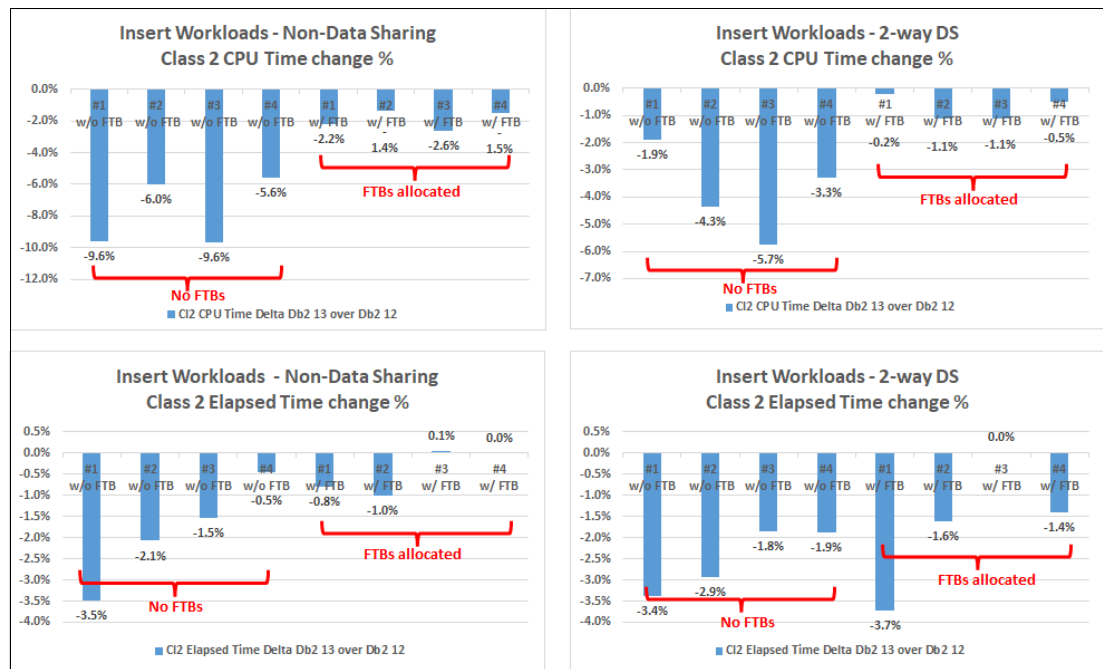


Figure 7-16 Insert workloads: Class 2 CPU and class 2 elapsed time changes (Db2 13 versus Db2 12)

- For the tests with FTBs allocated for all the indexes, we observed a total getpage reduction of 37% for all the indexes and a total class 2 CPU time saving of approximately 2%. For more information, see Figure 7-16.

**Note:** When FTBs are allocated for the indexes, we see fewer CPU savings because fast index traversal already helps to reduce the index tree traversal cost. The index look-aside function is further reducing CPU cost, in addition to fast index traversal, by performing fewer index leaf page getpage requests.



For the data-sharing tests, we observed the following results when comparing the Db2 13 performance results to the Db2 12 results:

- ▶ For the tests without FTBs allocated, a total getpage reduction of 30 - 41% for all the indexes (depending on whether **RUNSTATS** was run for the Db2 12 tests), we observed a reduction of 2 - 6% total class 2 CPU time.
- ▶ For the tests with FTBs allocated for all the indexes, we observed a total getpage reduction of 27 - 37% for all the indexes (depending on whether **RUNSTATS** was run for the Db2 12 tests), which resulted in a total class 2 CPU usage saving of approximately 1%.

**Note:** The data-sharing tests showed fewer CPU savings with the Db2 13 index look-aside enhancements than the non-data-sharing tests, even though they have similar reductions in getpage activity. This result is expected because in a data-sharing environment, the getpage cost makes up less of the cost of the overall transaction compared to a non-data-sharing environment.

We also checked the Db2 system address space CPU usage and storage usage, and no notable differences were observed between the Db2 12 and Db2 13 tests.

Figure 7-16 on page 256 depicts the class 2 CPU time and class 2 elapsed time changes for the insert workloads. The results from testing with Db2 13 are compared to the results of testing with Db2 12. For workload characteristics details for scenarios 1, 2, 3, and 4, see “Measurement scenario description ” on page 252.

### ***Update workload performance***

Before Db2 13, index look-aside was not supported for SQL update operations. The performance evaluations of the Db2 13 enhanced index look-aside support for update operations have two goals: to make sure there is no performance regression, and to make sure that the enhancement is providing the intended performance benefits.

To check for possible regression, we designed a random update workload, which is referred to as update workload scenario 1, which is described in “Measurement scenario description ” on page 252.

To evaluate whether index look-aside support for **UPDATE** operations is beneficial, we also created a sequential update workload, which is referred to as update workload scenario 2.

Analyzing the getpage activity of the indexes with a different cluster ratio for these update workloads shows a reduction in the getpage count for all indexes with index look-aside enabled:

- ▶ For the random update workload, the getpage activity of the clustering index IXPBR01 was reduced by approximately 19% when fast index traversal (FTB) is not used.
- ▶ For the random update workload, except for the clustering index IXPBR01, all indexes show a getpage reduction of 29% when fast index traversal is not used.
- ▶ For the sequential update workload, when fast index traversal is not used, getpage activity was reduced 47% - 85% for each index, depending on their clustering ratio. The higher the clustering ratio, the greater the getpage reduction.
- ▶ For both update tests, when FTBs are allocated, the getpage reduction percentages are slightly less for all the indexes than the reductions for the same indexes when fast index traversal is not enabled for them.
- ▶ The index getpage reductions are similar for the test runs that use a non-data-sharing and a data-sharing configuration.

Figure 7-17 shows the index getpage counts for the update tests that we ran comparing Db2 12 and Db2 13 that used the 2-way data-sharing configuration.

**Note:** In Figure 7-17, *w/o FTB* means that FTB structures are not used for the indexes. *W/FTB* means that FTB structures are used for the indexes.

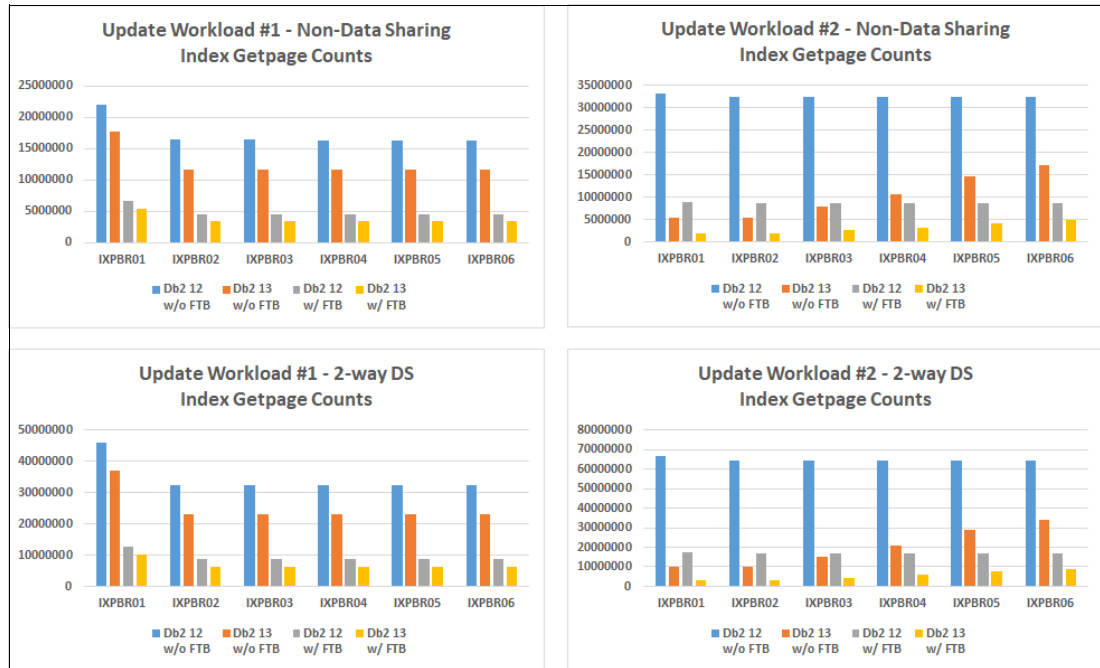


Figure 7-17 Update workloads: Total index getpage activity

With all indexes experiencing different levels of getpage reductions, the workload CPU usage also shows different degrees of savings. The sequential update tests that do not have fast index traversal that is enabled for the indexes show the largest class 2 CPU savings, with over 16% among all the tests.

The following list summarizes a few key observations about changes in CPU usage for the update workloads when comparing the Db2 13 runs to the Db2 12 runs, as shown in the top half of Figure 7-18 on page 259:

- ▶ For update workload 1 (random updates), more than 2% class 2 CPU saving was observed when fast index traversal is not used. The minor increase in class 2 CPU time for one of the tests in non-data sharing with FTB enabled for the indexes is small and within measurement noise level.
- ▶ The update workload 2 (sequential update) shows a class 2 CPU time savings of 16% when fast index traversal is not used in a non-data-sharing environment. In a 2-way data-sharing environment, the same test shows a class 2 CPU saving close to 12%.
- ▶ With fast index traversal enabled for the indexes, update workload 2 shows a class 2 CPU saving of 4% in a non-data-sharing environment, and 3% when using data sharing, which is good.
- ▶ The class 2 CPU time makes up less than 30% of the class 2 elapsed time for all the measurements, so the class 2 CPU time changes do not affect the class 2 elapsed time numbers much. As shown in the bottom half of Figure 7-18 on page 259, the class 2 elapsed time differences between the Db2 12 and Db2 13 tests are all within measurement noise level.

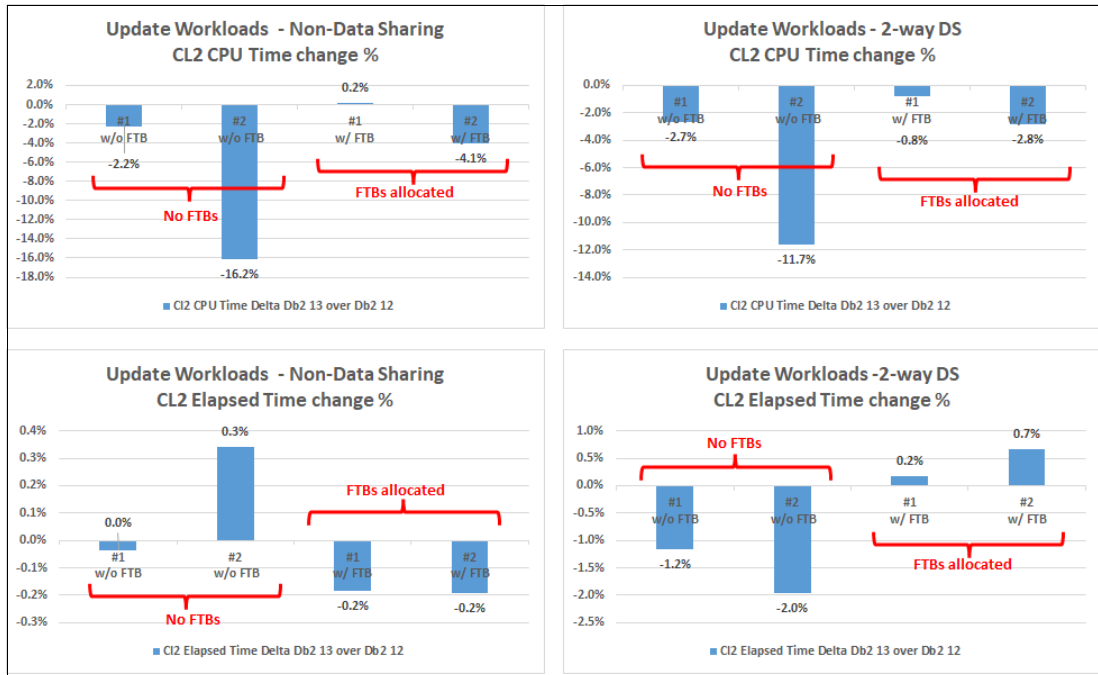


Figure 7-18 Update workloads: Class 2 CPU and class 2 elapsed time changes

We also verified the Db2 address space CPU and storage usage for these measurements, and we observed no notable differences between the Db2 12 and Db2 13 tests.

### Delete workload performance

For SQL delete operations, we expect to see the non-clustering indexes with low cluster ratios gain some benefits from the capability of dynamically enabling index look-aside in Db2 13. To validate this assumption, we designed a sequential delete workload with indexes that have different cluster ratios and measured the workload performance by using Db2 12 and Db2 13. For detailed workload characteristics, see “Measurement scenario description” on page 252.

Analyzing the performance results of the different test runs shows overall positive results for the Db2 13 tests.

The following list summarizes the differences in getpage activity for the indexes:

- ▶ All non-clustering indexes with a cluster ratio greater than 0 show a getpage count reduction from 15% to 84%. The greater the cluster ratio, the bigger the percentage of the reduction.
- ▶ The getpage counts for clustering index IXPBR01 are similar between Db2 12 and Db2 13 tests because index look-aside is used for this index in both Db2 12 and Db2 13.
- ▶ For index IXPBR06, which has a cluster ratio of 0, getpage counts are similar between the Db2 12 and Db2 13 tests because index look-aside does not help avoid getpages for this index.
- ▶ For all six indexes, without FTBs allocated, Db2 13 performs 39% fewer getpages for these indexes in total. With FTBs allocated, Db2 13 performs 34% fewer index getpages in total.

Figure 7-19 provides more details about the getpage reduction for the different indexes for the (sequential) delete workload in a non-data-sharing and data-sharing environment, with and without FTB enabled.

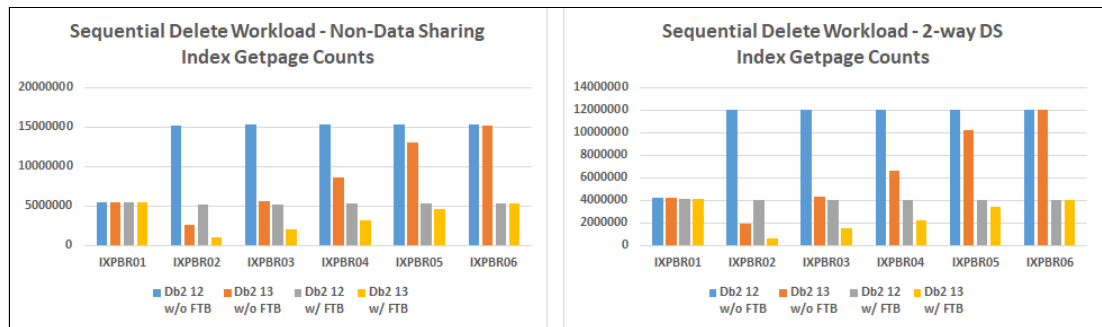


Figure 7-19 Delete workload: Total index getpages

As a result of the reduction in getpage activity, we see CPU time savings for three out of the four measurements, as shown in Figure 7-20:

- ▶ As with the insert and update test runs, we see more CPU savings from the Db2 13 enhanced index look-aside capability for the delete tests when fast index traversal is not enabled. We get more than an 8% class 2 CPU savings for the non-data-sharing test.
- ▶ When the sequential delete workload is run in a data-sharing environment, and all the indexes have FTBs that are allocated, there is a 0.4% class 2 CPU usage increase, which is small enough to be considered measurement noise.
- ▶ The Db2 class 2 elapsed time of the delete application was reduced for all the sequential delete scenarios with Db2 13, as shown at the right of Figure 7-20.

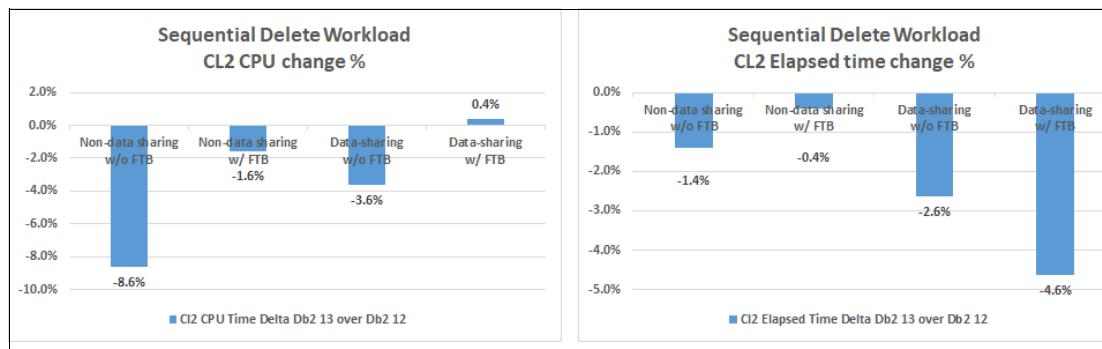


Figure 7-20 Delete Workload total class 2 CPU and total class 2 elapsed time changes

We also checked the Db2 address space CPU and storage usage for the delete workload measurements, and we observed no notable differences between the Db2 12 and Db2 13 tests.

## Summary of results

The results of this performance study show that the optimized index look-aside capability in Db2 13 can improve index insert, update, and delete efficiency for indexes with low cluster ratios in all types of workloads that were tested. With index look-aside being dynamically monitored and turned on and off for individual indexes, Db2 13 makes sure that look-aside is enabled only when it benefits an index and that it is turned off when there is no benefit.

Therefore, in the sequential insert, update, and delete workloads that were measured, regardless of whether the index cluster ratio is correctly reflected in catalog statistics, indexes with a nonzero cluster ratio all showed index getpage reductions. The ability of Db2 13 to quickly detect and deactivate ineffective index look-aside ensures that random type workloads do not waste processor time maintaining index look-aside information that is not helpful for an index.

Fast index traversal (FTB) is a powerful Db2 performance feature to reduce index getpages. When fast index traversal is in effect for an index, the index access is greatly optimized. Understandably, the performance benefits of using index look-aside for an FTB-enabled index are not as prominent as when index look-aside is used for an index that does not have FTBs allocated.

We also observed performance benefits from the Db2 13 enhanced index look-aside capability in other workloads that were not designed for evaluating index look-aside performance. Some of the sequential high-insert batch workloads (HIWs) also experienced notable CPU time savings with reduced index getpage activity. Some batch subworkloads in the performance regression bucket (PRB) workload experienced performance improvements from this enhancement too. For more information about HIW and PRB, see Chapter 4, “Workload-level measurements” on page 149.

## 7.3 Index management diagnostic and serviceability enhancements

Index split activity is one of the main contributors of insert elapsed time. Especially in a data-sharing environment, index split processing against a group buffer pool (GBP)-dependent index can cause a noticeable impact because the split process can trigger elongated Db2 latch or P-lock suspensions, and log force write waits. Occasionally, an index split operation encounters some situation where it might take a relatively long time to complete, which can cause insert activity to become noticeably slow. Diagnosing index split problems is difficult because it requires you to identify the potential problematic index, narrow down the time window, identify related units of recovery in the Db2 recovery logs, and so on.

To mitigate the overhead that is involved with diagnosing index split problems, Db2 13 introduces a new Instrumentation Facility Component Identifier (IFCID) 396 trace record, and added three new fields in the Db2 catalog to track index splits and abnormal index splits.

### 7.3.1 The new index split fields in SYSIBM.SYSINDEXSPACESTATS

Db2 13 tracks all index split operations in memory and externalizes the statistics to three new columns in the SYSIBM.SYSINDEXSPACESTATS catalog table: REORGTOTALSPLITS, REORGSPPLITTIME, and REORGEXCSPLITS. (The same three columns are also added to the corresponding SYSIBM.SYSIXSPACESTATS\_H temporal table.)

These new fields, which are described in Table 7-2, are available starting at catalog level V13R1M501, and are populated even before FL 501 is activated.

Table 7-2 New fields to track index splits in SYSIBM.SYSINDESSPACESTATS

Column name	Description
REORGTOTALSPLITS	Total number of index splits since the last reorganization or rebuild
REORGSPPLITIME	Total or aggregated elapsed time of index splits since last reorganization or rebuild
REORGEXCSPLITS	Total number of abnormal index splits (such as elapsed time > 1 s) since the last reorganization or rebuild

### 7.3.2 The new IFCID 396

A new IFCID 396 is included in statistics trace class 3 and performance trace class 6. Whenever an index split takes more than 1 second, and if you activated this trace (normally you should), Db2 creates an IFCID 396 record, which records information that can help you diagnose this long index split.

Testing showed that the cost of this trace is minimal, even if you have a large volume of exceptional index splits. In this case, the only thing you might notice is an increase in the trace volume for the trace destination that is used by IFCID 396. Alternatively, you can turn on IFCID 359, which records every event of index splits in Db2 12 and prior releases. However, IFCID 359 can become a large volume of trace data in heavy insert workloads and be cumbersome to use to identify the record with high wait time. The new IFCID 396 improves productivity by filtering on only problematic records.

IFCID 396 is available starting at Db2 13 FL 100, and contains the fields that are shown in Table 7-3.

Table 7-3 IFCID 396 fields

IFCID 396 field	Description
QW0396_Eye CHAR(8)	Eye catcher of IFCID396.
QW0396DBID CHAR(2)	Database ID of current splitting index.
QW0396PSID CHAR(2)	Page set ID of current splitting index.
QW0396MemID FIXED(16)	Data-sharing member ID of current splitting index. 0 means non-data sharing.
QW0396PartNum FIXED(16)	Partition number of current splitting index.
QW0396URID CHAR(10)	UR ID related with current splitting index.
Unused CHAR(2)	Reserved for future use.
QW0396PageNum FIXED(32)	Current splitting page number.
QW0396TimeStamp CHAR(10)	Current timestamp of IFCID 396 generation.
QW0396GBPD CHAR(1)	Whether the current index page set is marked as GBP dependent. 'Y' = GBP dependent.

IFCID 396 field	Description
Unused CHAR(1)	Reserved for future use.
QW0396ElapseTime FIXED(32)	Total elapsed time of current abnormal index split (in milliseconds).

### 7.3.3 Diagnosis of index split problems

The three new columns in the SYSIBM.SYSINDEXSPACESTATS catalog table can be used to provide a regular health check, identifying which indexes tend to have many splits, which indexes have more unusually long index splits, or both. It also can act as a baseline by calculating the average split time so that you can track this information over time.

When you think that insert activity is getting slow, and you suspect that it might be related to long or many index splits, you can use the following process to diagnose the situation:

1. Review the Db2 accounting data, and check whether Db2 latch suspension time is one of the major contributors and takes more time than normal, as shown in Figure 7-21.

You can also use the **TOP** command in the IBM Tivoli OMEGAMON for Db2 Performance Expert (OMPE) reporting tool to report transactions for the main contributors.

Display Filter View Print Options Search Help							
SDSF OUTPUT DISPLAY PE				JOB03143 DSID 128 LINE 33			
COMMAND INPUT ==>				COLUMNS 02- 133			
---				SCROLL ==> PAGE			
--- IDENTIFICATION ---							
ACCT TSTAMP:	09/21/21 22:04:21.53	PLANNAME:	db2jcc_a	WLM SCL:	DDFWORK	CICS NET:	N/A
BEGIN TIME :	09/21/21 22:04:20.18	PROD TYP:	JDBC DRIVER	LUM NET:	G91E81DA	CICS LUN:	N/A
END TIME :	N/P	PROD VER:	V4 R23M0	LUM LUN:	6C99	CICS INS:	N/A
REQUESTER :	::9.30.129.218	CORRNAME:	db2jcc_a	LUM INS:	DA59EAA9A003		
MAINPACK :	db2jcc_a	CORRNMBR:	ppli	LUM SEQ:	13228	ENDUSER :	USRT001
PRIMAUTH :	USRT001	CONNTYPE:	DRDA			TRANSACTION:	Insert_ORDERS
ORIGAUTH :	USRT001	CONNECT :	SERVER			WSNAME :	9.30.129.218
TIMES/EVENTS	APPL (CL.1)	DB2 (CL.2)	CLASS 3 SUSPENSIONS	ELAPSED TIME	EVENTS	TIME/EVENT	HIGHLIGHTS
ELAPSED TIME	4.856821	4.855962	LOCK/LATCH(DB2+IRLM)	4.631017	39	0.118744	THREAD TYPE : DBAT
NONNESTED	4.856821	4.855962	IRLM LOCK+LATCH	0.000000	0	N/C	TERM.CONDITION: NORMAL
STORED PROC	0.000000	0.000000	DB2 LATCH	4.631017	39	0.118744	INVOKE REASON : END USER
UDF	0.000000	0.000000	SYNCHRON. I/O	0.031954	35	0.000913	PARALLELISM : N/A
TRIGGER	0.000000	0.000000	DATABASE I/O	0.005300	12	0.000442	PCA RUP COUNT : 0
			READ CACHE HIT	0.002908	6	0.000485	RUP AUTONOM.PR: 0
CP CPU TIME	0.019624	0.019386	LOG WRITE I/O	0.026654	23	0.001159	AUTONOMOUS PR : 0
AGENT	0.019624	0.019386	OTHER READ I/O	0.009420	10	0.000942	QUANTITY : 10
NONNESTED	0.019624	0.019386	OTHER WRTE I/O	0.000615	1	0.000615	COMMITTS : 10
STORED PROC	0.000000	0.000000	SER_TASK SWTCH	0.000000	0	N/C	ROLLBACK : 0

Figure 7-21 Sample accounting trace record with high Db2 LATCH suspension time

- Review the Db2 statistics data, and check whether LC06 or LC254 is high because index-tree-split-related suspensions are recorded under LC06 or LC254 (see Figure 7-22). If so, the major contributor to the long Db2 LATCH suspensions in accounting is likely index split activity.

```
INTERVAL END : 09/21/21 22:05:00.47 TOTAL THREADS : 0
```

ACCOUNTING ROLLUP	QUANTITY	/SECOND	/THREAD	/COMMIT	LATCH CNT	/SECOND	/SECOND	/SECOND	/SECOND
ROLLUP THRESH RECS WRITTEN	9008	150.13	N/C	0.08	LC01-LC04	0.00	0.00	0.00	0.00
STORAGE THRESH RECS WRITTEN	0	0.00	N/C	0.00	LC05-LC08	0.02	4519.74	0.17	0.00
STALEN THRESH RECS WRITTEN	0	0.00	N/C	0.00	LC09-LC12	0.00	0.00	0.00	0.30
RECS UNQUALIFIED FOR ROLLUP	0	0.00	N/C	0.00	LC13-LC16	0.00	512.85	0.00	0.18
					LC17-LC20	0.00	0.00	0.02	45.75
					LC21-LC24	0.05	0.00	31.93	0.13
					LC25-LC28	0.00	0.00	0.43	0.00
					LC29-LC32	0.03	0.58	0.05	0.27
					LC33-LC36	0.00	0.00	0.00	0.00
					LC37-LC40	0.00	0.00	0.00	0.00
					LC41-LC44	0.00	0.00	0.00	0.00
					LC45-LC48	0.00	0.00	0.00	0.00
					LC49-LC52	0.00	0.00	0.00	0.00
					LC53-LC56	0.00	0.00	0.00	0.00
					LC57-LC60	0.00	0.00	0.00	0.00
					LC61-LC64	0.00	0.00	0.00	0.00
					LC254	448.17			

Figure 7-22 Sample statistic trace record showing high LC06 and LC254 contention

Starting in Db2 13, you can also print the IFCID 396 records during a slowdown period, which provide information about long (> 1 second) index split events. Using this information, you can tell which application or applications and which index or indexes contributed to the index split exceptions. IFCID 396 also provides the URID and page number, which can help you or the IBM service team to zoom in on the problem. A sample (formatted) IFCID 396 record is shown in Figure 7-23.

```
USRT001 SERVER DA59EAA99846 USRT001 9.30.129.218 Insert_ORDERS
USRT001 db2jcc_a DRDA 22:04:23.55504054 92922 8 396 INDEX SPLIT NETWORKID: G91E81DA LUNAME: GC83
DISTSERV pp1i N/P
```

```
-----
|
|
| INDEX SPLIT INFORMATION
|
| DATABASE ID : FALCDB01 PAGE NUMBER : 2
| PAGE SET ID : PKRUIXRO URID : X'000000000C4FA3FF54D4'
| DATA SHR MEMBER ID: 1 TOTAL ELAPSED TIME: 1.308000
| PARTITION NUMBER : 1 CURRENT TIME STAMP: 09/21/21 22:04:23.555012
| GBP DEPENDENT : NO
|
```

Figure 7-23 Sample IFCID 396 trace record

- Using the index name from the IFCID 396 trace, you can query the SYSIBM.SYSINDEXSPACESTATS catalog table for this problem index, confirming the number of splits, the time they took, and the number of problematic splits. In our example, which is shown in Figure 7-24, we are evaluating index FALCDB01.PK\_UIX\_ORDER.

	DBNAME	NAME	INDEXSPACE	REORGTOTALSPLITS	REORGSPPLITTIME	REORGEXCSPLITS
1_	FALCDB01	NEW_UIX_ORDE	NEWRUIXR	147413	2036034	994
2_	FALCDB01	PK_UIX_ORDER	PKRUIXRO	421393	5859004	2812
3_	FALCDB01	UIX_ORDERS	UIXRORDE	246423	1928695	957

Figure 7-24 Sample of catalog query output on SYSIBM.SYSINDEXSPACESTATS



4. After the problematic index is identified, you can explore some tuning options or actions for the index, the application, or both, depending on the insert pattern. For example, if the insert pattern is sequential, and if the index page size is less than 32 K, then increasing the index page size can reduce the number of splits by fitting more keys in a single page. If the insert is mostly random, increasing PCTFREE and performing the REORG more frequently to leave more space for future inserts reduces the need for index splits.

## 7.4 Application timeout and deadlock control

In earlier Db2 versions, the **IRLMRWT** subsystem parameter governs the timeout interval for most lock requests, which prevents you from having more granular control to set different timeout values for specific applications. In addition, certain Data Definition Language (DDL) activities can be prone to deadlocks. If the thread performing the DDL is chosen as the deadlock victim, the scheduled DDL activities fail, which requires more attempts that can impact subsequent work. Db2 13 enhances the support and flexibility to allow you and your applications to assign values to the new **CURRENT LOCK TIMEOUT** special register and to the **SYSIBMADM.DEADLOCK\_RESOLUTION\_PRIORITY** built-in global variable.

The **CURRENT LOCK TIMEOUT** special register can be set at the application level to control the number of seconds that the application waits for a lock before it times out. Db2 13 must be at FL 500 and later, and the application must be set up to use application compatibility (APPLCOMPAT) V13R1M500 or later.

With the **SYSIBMADM.DEADLOCK\_RESOLUTION\_PRIORITY** built-in global variable, you can specify a deadlock resolution priority value for internal resource lock manager (IRLM) to use when resolving deadlocks with other concurrent threads. This feature requires Db2 13 to be at FL 501 or later, and the application to use APPLCOMPAT (V13R1M501) or later. Db2 13 FL 501 is required because the new built-in global variable is added with catalog level V13R1M501.

To leverage this new function, install the PTF for IRLM APAR PH43770.

### 7.4.1 Performance measurements

To verify that no performance regression occurs when using these new features, several performance tests were conducted. The basic performance regression suite and distributed IBM Relational Warehouse Workload (IRWW) workloads were run with and without these enhancements.

#### Results

No performance regression was observed for either workload. For more information about these workloads, see Appendix A, “IBM Db2 workloads” on page 403.

### 7.4.2 Monitoring information

To monitor the **CURRENT LOCK TIMEOUT** special register, new counters were added to the accounting and statistics trace record to report how often the **SET CURRENT LOCK TIMEOUT** statement is used in both accounting and statistics traces.

In addition, a new trace record, IFCID 437, was added to record every use of the **SET CURRENT LOCK TIMEOUT** statement.

New fields were added to IFCID 196 (timeout) and IFCID 172 (deadlock) to indicate whether the timeout, deadlock, or both priority values are set by the **CURRENT LOCK TIMEOUT** special register and the **DEADLOCK\_RESOLUTION\_PRIORITY** global variable.

For more information about these instrumentation changes, see 12.6, “IFCID changes for application timeout and deadlock control” on page 357.

### 7.4.3 Usage considerations

Using the **SET CURRENT LOCK TIMEOUT** special register and the **DEADLOCK\_RESOLUTION\_PRIORITY** built-in global variable requires application changes. To make it easier to implement these changes, Db2 13 extends the support of profile tables so that you can specify the **SET CURRENT LOCK TIMEOUT** special register and **DEADLOCK\_RESOLUTION\_PRIORITY** built-in global variable by using Db2 profile tables for both the local and distributed applications. For more information about this enhancement, see 7.5, “Enhanced support for profile tables” on page 266.

## 7.5 Enhanced support for profile tables

Before Db2 13, special registers and system built-in global variables can be defined by using Db2 profile tables, but only for distributed threads. In Db2 13, Db2 profile table support is enhanced to allow the specification of the new **SET CURRENT LOCK TIMEOUT** special register and the new **DEADLOCK\_RESOLUTION\_PRIORITY** built-in global variable for both local and remote applications if the system is at a sufficient FL and the application is bound with an **APPLCOMPAT** value that supports the usage of the new special register and global variable. (These requirements are described in 7.4, “Application timeout and deadlock control” on page 265.)

In addition, you can also specify a new **RELEASE\_PACKAGE** keyword in the **DSN\_PROFILE\_ATTRIBUTES** profile table to change the **RELEASE** bind option of a package from **DEALLOCATE** to **COMMIT** behavior so that the concurrent DDL statements can break in. The new **RELEASE\_PACKAGE** keyword also applies to both local and remote applications.

For local threads, during the period that the profile is active, each time a local package is loaded for running, Db2 performs a look-up in the profile tables to find the correct profile based on the filtering criteria that is specified in **DSN\_PROFILE\_TABLE**. A package is loaded when the first SQL statement in that package runs. If there is a match of the criteria, Db2 performs the specified actions in **DSN\_PROFILE\_ATTRIBUTES** for the matching profile, for example, **SET CURRENT LOCK TIMEOUT = 200**. If the package is released at **COMMIT** due to the **RELEASE (COMMIT)** bind option, the next SQL statement from that package that runs after the **COMMIT** loads the package again.

For remote threads, profiles are evaluated and **SET** statements are processed only when the first package loads, and the first non-**SET** statement within that package runs. Subsequent package loads do not cause the profile to be re-evaluated.

At the time of writing, only the use of **SET CURRENT LOCK TIMEOUT**, **DEADLOCK\_RESOLUTION\_PRIORITY**, and **RELEASE\_PACKAGE** options are supported for local applications. No other special registers or global variables are supported for local profiles.

The following requirements must be met before you can leverage these profile enhancements:

- ▶ Db2 13
- ▶ FLs:
  - For **RELEASE\_PACKAGE**: V13R1M500 or later
  - For **SET CURRENT LOCK TIMEOUT RELEASE\_PACKAGE**: V13R1M500 or later
  - For **DEADLOCK\_RESOLUTION\_PRIORITY**: V13R1M501 or later

The system profile table support for local or remote applications requires Db2 to be started with the DDF subsystem parameter set to AUTO or COMMAND.

## 7.5.1 Performance measurements

The following performance tests were designed to verify that minimal and reasonable CPU time overhead occurs when using system profiles for local threads.

### Test case 1

This test case measures the CPU time overhead by using a simple, single-thread query:

```
SELECT 1 FROM SYSDDUMMY1
```

The application package is bound with **RELEASE(COMMIT)**. It repeats the same query 100 times and commits after each query. Both **SET CURRENT LOCK TIMEOUT** and **SYSIBMADM.DEADLOCK\_RESOLUTION\_PRIORITY** are specified in the profile tables.

Before the test starts, a distributed Java program populates the Db2 internal hash table that stores the profiles with 50,000 different connections, each with a distinct Db2 application name.

The following test scenarios are used:

- ▶ Profiling is active with matching profile attributes.
- ▶ Profiling is active with non-matching profile attributes.
- ▶ Profiling is deactivated.

### Test case 2

This test case measures the CPU time overhead with a long-running query with a single table access that returns 15,000 rows, and then issues a commit that is followed by issuing 100 searched **UPDATE** statements with a commit after each **UPDATE** statement. The target updated table has 4.5 million rows. The application package is bound with **RELEASE(COMMIT)**. The **SET CURRENT LOCK TIMEOUT** statement is specified in the profile table. The following test scenarios are used:

- ▶ Profiling is active with matching profile attributes.
- ▶ Profiling is active with non-matching profile attributes.
- ▶ Profiling is deactivated.

### Test case 3

This test case evaluates the CPU time overhead and successful DDL break-in rate for local profile support that specifies the **RELEASE\_PACKAGE** option. We use the distributed IRWW 400 Warehouse (32 GB) Online Transaction Processing (OLTP) workload for this test. The workload calls different read/write transactions that are implemented by using native SQL stored procedures. Package names of the native stored procedures are used in `SYSIBM.DSN_PROFILE_TABLE` as the matching criteria. The detailed workload description can be found in Appendix A, “IBM Db2 workloads” on page 403.

The following extra setup is used for this evaluation:

- ▶ The **RELEASE\_PACKAGE** profile attribute is set to `COMMIT`.
- ▶ The `ATTRIBUTE2` column in `SYSIBM.DSN_PROFILE_ATTRIBUTES` is set to 1, which indicates that this profile keyword applies to local threads. Because the SQL statements that are run as part of the stored procedure packages do not originate directly from a remote client, they are considered local, so they qualify for `ATTRIBUTE2 = 1 (local)`.
- ▶ Three out of seven native SQLPL packages match the profile attribute.
- ▶ Single-thread **ALTER TABLE** statements are issued every 5 seconds in a 10-minute interval through a separate batch job.
- ▶ The SQLPL packages are bound with **RELEASE (DEALLOCATE)**.

The following test scenarios are used:

- ▶ Profiling is active with matching profile attributes.
- ▶ Profiling is deactivated.

### Test case 4

This test case evaluates the CPU time overhead and successful DDL break-in rate by using the distributed SQLJ IRWW 400 Warehouse (32 GB) OLTP workload for remote profile support. The detailed workload description can be found in Appendix A, “IBM Db2 workloads” on page 403.

The following additional setup is used for this evaluation:

- ▶ The **RELEASE\_PACKAGE** profile attribute is set to `COMMIT`.
- ▶ The `ATTRIBUTE2` column in `SYSIBM.DSN_PROFILE_ATTRIBUTES` is set to `NULL`, which indicates that this profile keyword applies to distributed threads.
- ▶ Three out of seven SQLJ packages match the profile attribute.
- ▶ Single-thread **ALTER TABLE** statements are issued every 5 seconds in a 10-minute interval through a separate batch job.
- ▶ The SQLJ packages are bound with **RELEASE (DEALLOCATE)**.

The following three test scenarios are measured with and without the new remote profile enhancement implementation (feature on versus feature off):

- ▶ Profiling is active with matching profile attributes with **DDF PKGREL BNDOPT**.
- ▶ Profiling is deactivated with **MODIFY DDF PKGREL** set to `COMMIT`.
- ▶ Profiling is deactivated with **MODIFY DDF PKGREL** set to `BNDOPT`.

### Measurement environment

The performance measurements for test case 1 and test case 2 were conducted by using an IBM z14 with six general-purpose central processors (GPCs), two ZIIPs with 80 GB of memory, and DS8800 disk storage.

Test case 3 and test case 4 used the following setup:

- ▶ An IBM z15 z/OS LPAR with three dedicated general CPs, two zIIP CPs, and 32 GB of memory
- ▶ z/OS 2.5
- ▶ Linux on IBM Z on an IBM z15 LPAR with three general CPs with Db2 for Linux, UNIX, and Windows 11.5
- ▶ IBM Data Server Driver for JDBC and SQLJ 4.31.10

## Measurement results

The measurements from test case 1 revealed the following key results:

- ▶ Profiling activated with matching attribute versus profiling deactivated  
Up to 10% class 2 CPU time overhead was observed with a difference in absolute numbers of about 100 ms.
- ▶ Profiling activated with non-matching attribute versus profiling deactivated  
Up to 5% class 2 CPU time overhead was observed with a difference in absolute numbers of about 60 ms.

The measurements from test case 2 indicated that due to the complexity of the SQL statements that are injected by the application, the class 2 CPU overhead is within the noise range for both comparisons (that we also used for test case 1).

The measurements from test case 3 revealed the following key results:

- ▶ An average class 2 CPU time overhead around 1.37% at the (complete) workload level when comparing profiling activated with profiling deactivated
- ▶ The successful DDL break-in rate increased from 45% (54 out of 120) with profiling deactivated to 100% with profiling activated.

For test case 4, the results of the different measurement scenarios are shown in Table 7-4. “Feature off” means that the Db2 code to support this enhancement was not installed, but “Feature on” means that the code for this enhancement was installed.

Table 7-4 Performance results of using `RELEASE_PACKAGE` profile support with remote applications

Profile and PKGREL		DDL successful break-in rate (%)		Total CPU time in microseconds/commit	
Profile	PKGREL	Feature off	Feature on	Feature off	Feature on
Off	COMMIT	100	100	285	285
Off	BNDOPT	0	0	204	204
ON and matching profile attributes	BNDOPT	0	100	208	292

The measurements in Table 7-4 show that enabling high-performance database access threads (DBATs) (`PKGREL BNDOPT` to trigger `RELEASE(DEALLOCATE)`) behavior reduces the average CPU time per transaction in this particular workload (comparison of first two rows).

However, with high-performance DBATs, the application thread that performs DDL operations cannot break in, and timeouts occur when the system profile is not turned on.

With the new remote profile enhancement, the thread that is performing the DDL operations can break in and successfully run the DDL statements when the system profile is on (with a matching profile that specifies the **RELEASE\_PACKAGE** keyword). Because the packages are changed to **RELEASE(COMMIT)** behavior by using this feature, the benefits of high-performance DBATs are unavailable, as indicated in the CPU time in the last row in the table. For this reason, it is a best practice that you turn on a profile only when you must, and after achieving the goal (DDL completed successfully), you turn off the profile again.

### Summary of results

More detailed analysis of the CPU time showed that the extra CPU time overhead comes from Db2 loading the package, matching the profile, and performing the specified action in the profile tables. The percentage of the CPU time overhead can be workload-dependent. Db2 system profiles are efficient and easy to use and can help you to temporarily change existing application behavior to achieve a higher level of application concurrency without modifying existing applications.

## 7.5.2 Usage considerations

Consider the following best practices for using system profiles:

- ▶ To minimize the cost of non-matching lookups in the **SYSIBM.DSN\_PROFILE\_TABLE**, enable the profile function only when needed. You can do so by setting the **PROFILE\_ENABLED** value to 'N' and issuing a **-START PROFILE** command to refresh the in-memory structures that contain the profile information. You also can stop all profile activity by using the **-STOP PROFILE** command if system profiles are used only to override the **RELEASE(DEALLOCATE)** option by using the **RELEASE\_PACKAGE** keyword.
- ▶ Periodically maintain and clean up out-of-date profile table entries.

## 7.5.3 Conclusion

A slight performance overhead is incurred when using system profiles because Db2 must search the internal hash table to find the matching profile and apply the specified action or actions in the **DSN\_PROFILE\_ATTRIBUTE** table at package load time. The percentage of the CPU time overhead is workload-dependent. Generally, the more complex the statements in a package, the smaller the observed overhead percentage.

Using system profiles provides flexibility and an easy way for you to change the application behavior temporarily without paying the cost of modifying the existing applications to achieve a higher level of concurrency in some circumstances, such as running concurrent DDLs.



## Query performance

Db2 13 introduces the following Db2 runtime performance enhancements that are related to query processing:

- ▶ Sort optimization for Db2 built-in functions with multiple DISTINCT, GROUPING SETS, and PERCENTILE clauses
- ▶ SUBSTR support for the LISTAGG function
- ▶ Improving ORDER BY sorts for long VARCHAR columns
- ▶ APREUSE storage reduction

## 8.1 Sort optimization for Db2 built-in functions with multiple **DISTINCT**, **GROUPING SETS**, and **PERCENTILE** clauses

In Db2 13, there are enhanced sort-intensive queries that involve Db2 built-in functions that contain multiple **DISTINCT** (two or more) clauses, **GROUPING SETS** (**GROUPING SETS**, **ROLLUP**, or **CUBE**) and **PERCENTILE** (**PERCENTILE\_DISC** or **PERCENTILE\_CONT**) clauses.

### 8.1.1 Requirements

To use this feature, you must be using Db2 13 at function level (FL) 100 or later. No rebind is needed.

### 8.1.2 Performance measurements

To evaluate this feature, a set of performance measurements were conducted by using Db2 13 with and without the performance enhancement to achieve the following goals:

- ▶ To verify the CPU time improvement for qualified queries
- ▶ To verify the reduction in sort buffer pool usage

#### Measurement environment

The performance evaluation was conducted on IBM z14 hardware with four general processors and three IBM zSystems Integrated Information Processors (zIIPs), with 32 GB of storage, and running z/OS 2.4 and Db2 13. IBM DS8800 (disk) storage devices were used.

#### Measurement scenario description

A number of **SELECT** queries that use qualified Db2 built-in functions with multiple **DISTINCT**, **GROUPING SET**, or **PERCENTILE** clauses were used during this evaluation, as shown in Example 8-1.

*Example 8-1 Queries that can benefit from Db2 13 sort enhancements*

---

```
SELECT C8,C9,C16,C32,
SUM(INT1),SUM(DECM)
FROM SORTTAB3
WHERE ROWNUM <= 10000
GROUP BY GROUPING SETS ((C8,C9),
                        (C16,C32)) ;

SELECT C16,
SUM(DISTINCT INT1),SUM(DISTINCT INT2),COUNT(*)
FROM SORTTAB3
WHERE ROWNUM <= 10000
GROUP BY C16 ;

SELECT C128, PERCENTILE_DISC(0.5)
WITHIN GROUP (ORDER BY INT1 )
FROM SORTTAB3
WHERE ROWNUM <= 10000
GROUP BY C128;
```

---



## Results

The following improvements were observed:

- ▶ Class 2 CPU usage for the multiple **DISTINCT** case was reduced by approximately 4%.
- ▶ Class 2 CPU usage for the **GROUPING SETS** case was reduced by up to 40%.
- ▶ Class 2 CPU usage for the percentile case was reduced by up to 12%, with a reduction in getpage activity for the sort work file buffer pool.

## Summary of results

Performance measurements showed up to a 40% reduction in class 2 CPU time and up to a 67% reduction in the number of getpage requests for the sort work file buffer pool for qualified queries. Generally, the more the data that is in sorting order when it arrives in the sort component, the more CPU time savings that can be expected.

### 8.1.3 Conclusion

Workloads with heavy usage of qualified queries can expect a reduction in both CPU time and number of getpage requests for sort buffer pool activity.

## 8.2 SUBSTR support for the LISTAGG function

In Db2 13, when the **LISTAGG** built-in function is used as the first argument for **SUBSTR** function and both the start and length of the **SUBSTR** are literal values, the Db2 sort component can condense the internal result buffer of the **LISTAGG** function from the default of 4000 bytes to the size of the **SUBSTR** setting. For example, if a query uses **SUBSTR(LISTAGG..., 1, 20)**, instead of allocating a sort result buffer size of 4000 bytes, Db2 allocates the buffer with a size of 20 bytes. This optimization results in the reduction of the work file usage and improves the performance of the **SUBSTR** function.

### 8.2.1 Requirements

To use this feature, you must be using Db2 13 at FL 100 or later. No rebind is required.

### 8.2.2 Performance measurements

To evaluate this feature, a set of performance measurements were conducted by using Db2 13 with and without the performance enhancement to achieve the following goals:

- ▶ To verify the CPU time and elapsed time improvement with qualified queries
- ▶ To verify the reduction in sort buffer pool usage

#### Measurement environment

The performance measurements were conducted on IBM z14 hardware with four general processors, three zIIPs, 32 GB of storage, and running z/OS 2.4 and Db2 13. DS8800 (disk) storage devices were used.

## Measurement scenario

A number of **SELECT** queries that use the **SUBSTR** and **LISTAGG** functions were used to evaluate this enhancement. The queries used different **SUBSTR** start and length input literal values and a varying number of rows to be sorted. A sample query is shown in Example 8-2.

*Example 8-2 SUBTR with LISTAGG sample query*

---

```
SELECT SUBSTR(LISTAGG(C1, ','),1,100) FROM T1
```

---

## Results

The following improvements were observed:

- ▶ The number of sort buffer pool getpage requests were reduced by 12% - 50%.
- ▶ Db2 class 2 CPU time was reduced by 11% - 60%.
- ▶ Up to 40% class 2 elapsed time reduction was observed.

## Summary of results

Performance measurements showed up to a 60% class 2 CPU time reduction and up to a 50% reduction in the number of getpage requests for the sort buffer pool for qualified queries. The more **SUBSTR** internal buffer storage savings, the more CPU time and elapsed time savings. The use of the **SUBSTR** function also reduces the size of the record that is written to the logical work file that is stored in the local buffer pool. This reduction allows for more entries to be stored on a single page and contributes to the reduction in getpage activity.

## 8.2.3 Conclusion

Workloads with heavy usage of qualified queries are expected to see reductions in both CPU time and elapsed time, and a reduction in the number of getpage requests for the sort work file buffer pools.

## 8.3 Improving ORDER BY sorts for long VARCHAR columns

In general, if the last column in a sort key is a long VARCHAR column, the sort component allocates storage for the maximum column length that is defined. In Db2 13, **SORT** records the maximum length that it encounters for the long VARCHAR column when the query is run the first time, and when it is an **ORDER BY** sort and the defined column length is greater than 100 bytes. When the same query runs again, **SORT** allocates an amount of storage based on the maximum actual data length that was observed and remembered from the first run. As a result, Db2 can reduce both the sort storage and the work file usage in subsequent executions of the same query.

### 8.3.1 Requirements

To use this feature, you must be using Db2 13 at FL 100 or later. No rebind is required.

## 8.3.2 Performance measurements

Performance measurements were conducted by using Db2 13 with and without the performance enhancement to achieve the following goals:

- ▶ To verify the CPU time and elapsed time improvements for qualified queries
- ▶ To verify the reduction in sort buffer pool usage

### Measurement environment

The performance measurements were conducted on IBM z14 hardware with four general processors, three zIIPs, 32 GB of storage, and running z/OS 2.4 and Db2 13. DS8800 (disk) storage devices were used.

### Measurement scenario

**SELECT** queries that include an **ORDER BY** clause and a defined length of the last column in the sort key that is greater than 100 bytes were used to evaluate this enhancement. The queries use different actual data sizes that are smaller than the defined column length. In Example 8-3, the **VARCH1** column is the candidate column that should be able to leverage this performance enhancement if its defined length is greater than 100 bytes.

*Example 8-3 Sample long VARCHAR sort query*

---

```
SELECT CHAR1, CHAR2, VARCH1
FROM T1
ORDER BY CHAR1, VARCH1;
```

---

### Results

The following improvements were observed for the second and subsequent run of the qualified queries:

- ▶ Class 2 CPU time was reduced by up to 10%.
- ▶ The number of sort buffer pool getpage requests was reduced by up to 24%.

### Summary of results

In general, the greater the difference between the actual data length and the defined length, the more memory storage and work file usage reduction for qualified queries can be expected, which results in more CPU time and elapsed time savings.

### Usage considerations

If the data in the table is updated or inserted with longer column values for the long VARCHAR column than Db2 recorded during the previous run, for the next run of the same query, the Db2 sort component detects this update and uses the new larger VARCHAR values, which can cause performance to degrade compared to the previous time that the query ran. If this situation happens, the performance enhancement is disabled for this query. However, the performance of the following query run should be the same as the first time that the query ran.

### 8.3.3 Monitoring information

The Db2 statistics counter ZAI SORT FEEDBACK USED (QXSTMLSRT), which is shown in Figure 8-1, reports how many times a query at the query block level leverages the information that is collected by the **Sort** component from the first time that the query runs at the system level. This information is collected with Db2 13 without IBM Db2 AI for z/OS (Db2ZAI) activated.

MISCELLANEOUS	
-----	
HIGH LOG RBA	0000000002E764E13EE4
BYPASS COL	0.00
MAX SQL CASCADING LEVEL	0.00
MAX STOR LOB VALUES (MB)	0.00
MAX STOR XML VALUES (MB)	0.00
ARRAY EXPANSIONS	0.00
SPARSE IX DISABLED	0.00
SPARSE IX BUILT WF	0.00
NO DM CALL RIDL/LPF	0.00
FETCH 1 BLOCK ONLY	0.00
RDS SORT PERFORMED	329.00
RDS SORTL USED	0.00
ZAI STABILIZED PREPARE	0.00
ZAI SORT FEEDBACK USED	96524.00
FTB THRESHOLD	1000.00

Figure 8-1 Db2 statistics report formatted by IBM OMEGAMON for Db2 Performance Expert

## 8.4 APREUSE storage reduction

When an application changes or database objects are altered, packages that depend on them are invalidated. In this case, a **REBIND** or a **BIND** is needed. You can rebind these invalidated packages manually or you can rely on the automatic rebind process that occurs the next time that the application runs.

During **REBIND**, the Db2 optimizer evaluates all the possible access plans to find the most optimal plan. This evaluation process requires virtual and real memory to store Db2 internal data structures for costing different join sequences. The larger the number of tables that must be joined, the more memory that is needed.

When the **APREUSE(ERROR)** or **APREUSE(WARN)** option is specified, Db2 first tries to retrieve the saved access plan by using a plan hint, and attempts to reuse the saved plan. If the access path is reused successfully, Db2 Query optimizer stops further processing. If Db2 fails to reuse the saved plan, the query optimizer falls back to using the normal cost evaluation process.

In Db2 12, **APREUSE(ERROR)** or **APREUSE(WARN)** processing always allocated storage to accommodate for the maximum number of tables that were joined and predicates in a query block.

In Db2 13, the Db2 query optimizer is enhanced to reduce the memory allocation for internal data structures to only the amount that is needed to apply the plan hint. By doing so, Db2 storage usage is reduced when the access path can be reused successfully. In addition, the Db2 optimizer also enhanced the memory allocation that is needed when Db2 fails to reuse the saved plan and must fall back to the general costing flow. Storage is allocated for only those structures that are affected by recosting the query.

Because the size of the specific Db2 data structures that are allocated is based on the complexity of the query (such as the number of tables and number of the predicates), we expect that more tables are joined in one query block, and the more complex the query, the greater the reduction in memory storage.

### 8.4.1 Requirements

To use this feature, you must be using Db2 13 at FL 100 or later.

### 8.4.2 Performance measurements

This section describes the measurements that were taken to assess the performance effects of the enhancements on the Db2 optimizer in relation to the **APREUSE** bind option.

#### Measurement environment

The performance measurements were conducted on IBM z14 hardware with four general processors, three zIIPs, 32 GB of storage, and running z/OS 2.4 and Db2 13. DS8800 (disk) storage devices were used.

#### Measurement scenarios

The measurements compare the **REBIND** performance of static packages that have queries with a different number of tables that are joined. The results are compared with and without the feature enabled.

- ▶ Scenario 1 measures a successful **APREUSE** with different numbers of joined tables with **APREUSE(WARN)**. Each measurement was run with 10 concurrent threads.
- ▶ Scenario 2 includes three different **APREUSE** scenarios for rebinding a package with a query that joins 200 tables in one query block. Each measurement is a single thread. The three **APREUSE** scenarios that were compared are as follows:
  - REUSE SUCCESS: **REBIND APREUSE(WARN)** and the access path was reused successfully.
  - REUSE FAIL: **REBIND APREUSE(WARN)** and the access path was not reused. A warning was issued.
  - REUSE NO: **REBIND APREUSE(NO)**.

#### Measurement results

The measurement results for both scenarios are provided in the following subsections.

### ***Different join number in one query block (scenario 1)***

The Db2 statistics report for these measurements showed a storage reduction of up to 26% in the 2 GB above-the-bar (ATB) shared real storage. As the number of tables that are joined increases, greater storage reductions were observed, as shown in Table 8-1.

There is no significant difference in both elapsed time and CPU time.

*Table 8-1 2 GB above-the-bar shared real storage savings for different number of tables that are joined*

<b>Number of tables joined</b>	<b>Old design (MB)</b>	<b>New design (MB)</b>	<b>Savings (MB)</b>	<b>Saving %</b>
200	553.2115	409.508	143.7035	-26.00%
17	192.265	172.423	19.842	-10.30%
2	53.1555	51.722	1.4335	-2.70%
Single table	57.106316	56.4955	0.610816	-1.10%

### ***Different APREUSE results (scenario 2)***

Table 8-2 shows a storage reduction of up to 29% when **REBIND** with **APREUSE** is successful, up to a 10% storage reduction when **APREUSE** failed, and an 8% reduction when **APREUSE** is not used.

*Table 8-2 2 GB above-the-bar shared real storage savings for different APREUSE results*

<b>Reuse result</b>	<b>Old design (MB)</b>	<b>New design (MB)</b>	<b>Savings (MB)</b>	<b>Saving %</b>
REUSE SUCCESS	53.9665	38.203	15.7635	-29.20%
REUSE FAIL	126.4115	112.9115	13.5	-10.70%
REUSE NO	128.8605	118.0005	10.86	-8.40%

## **Summary of results**

Successful **APREUSE** usage for queries in which many tables are joined in one query block greatly benefit from this enhancement. The storage reduction can be verified by looking at the 2 GB ATB shared real storage counters in the Db2 statistics report.

## **8.4.3 Conclusion**

The memory storage reduction for a successful **APREUSE** of static packages, especially ones with complex queries that involve many tables that are joined in one query block, helps overall system performance when running many **REBIND** operations concurrently as other Db2 applications in a memory-constrained system.



## IBM Db2 for z/OS utilities

This chapter describes the utility enhancements and features to reduce elapsed time and CPU and to improve performance that were made available after the initial release of Db2 12 and with Db2 13:

- ▶ LOAD PRESORT
- ▶ Online conversion from partition-by-growth to PBR
- ▶ Page sampling support for inline statistics
- ▶ Utility history
- ▶ Redirected recovery
- ▶ Migrating a multi-table table space to PBG universal table spaces
- ▶ REORG INDEX NOSYSUT1

## 9.1 LOAD PRESORT

Before this feature was introduced, loading a table in the clustering order required you to either sort the source data in your own job or step before the **LOAD**, or to perform a **REORG** after the **LOAD**. Db2 now supports a **PRESORT** keyword for the **LOAD** utility. This feature combines sorting the source data in clustering order and loading the table for you. By combining the steps, this feature provides both convenience in usage and potential improvements in elapsed time and CPU.

The following two extra enhancements are included in this feature to further improve **LOAD PRESORT** performance:

- ▶ The usage of the block-level interface both to and from IBM Db2 Sort for z/OS (Db2 Sort) instead of row by row to save CPU time when Db2 Sort is used. This enhancement applies to Db2 Sort only.
- ▶ Skipping the index key sort if the keys of an index match the clustering keys because the data is in order.

### 9.1.1 Requirements

To use this feature, your Db2 environment must meet the following minimum requirements:

- ▶ Db2 12 at function level (FL) 500 or later or Db2 13
- ▶ For Db2 12, APARs PH23105 and PH34323

### 9.1.2 Performance measurements

This section describes the measurements that were taken to assess the performance effects of the **LOAD PRESORT** utility enhancements.

#### Measurement environment

The performance evaluation was conducted on IBM z14 hardware with four general processors, one IBM zSystems Integrated Information Processor (zIIP), and running z/OS 2.3 and Db2 12.

#### Measurement scenario description

The table to load is a partition-by-range (PBR) table space with 30 partitions and five indexes (one partitioned index (PI), and four non-partitioned indexes (NPIs)).

The total number of records to load is approximately 180 million.

The following three scenarios were used to evaluate the performance improvements for the **LOAD PRESORT** utility enhancements:

- ▶ Scenario 1 does a **LOAD** followed by **REORG**. **LOAD** is using **INDEXDEFER ALL**, and **REORG** is using **SORTDATA**.
- ▶ Scenario 2 sorts the input source data in a separate job by using **DFSORT**, and then uses **LOAD** with **PRESORTED YES** specified.
- ▶ Scenario 3 uses **LOAD PRESORT**.



In each scenario, a measurement is taken for each of the following four combinations:

- ▶ The input source data is a single data set. Sorting uses **DFSORT**.
- ▶ The input source data is 30 data sets, with one data set for each partition so that intra-partition parallelism is used. Sorting uses **DFSORT**.
- ▶ The input source data is a single data set. Sorting uses Db2 Sort.
- ▶ The input source data is 30 data sets, with one data set for each partition so that intra-partition parallelism is used. Sorting uses Db2 Sort.

## Results

The performance comparison is between Db2 12 with the feature and Db212 without the feature.

The metrics for comparison are total elapsed time, CPU time, and zIIP time, which are shown in Figure 9-1.

	Scenario 1				Scenario 2				Scenario 3		
	LOAD followed by REORG				Sort Input then LOAD				LOAD PRESORT		
	LOAD Seconds	REORG Seconds	Total		Seconds	Sort input Seconds	LOAD Seconds		Total	Seconds	Seconds
DFSORT				Elapsed				Elapsed			
	202.1	140.6	342.7	CPU	161.0	280.3	441.3	CPU	318.6	-7%	-28%
	15.8	418.7	434.5	zIIP time	122.0	264.0	386.0	zIIP time	375.6	-14%	-3%
	154.1	94.0	248.1	CPU+zIIP	0.0	207.2	207.2	CPU+zIIP	251.5	1%	21%
		682.6				593.2		627.2	-8%	6%	
30 input source datasets (one per partition)	91.6	139.9	231.5	Elapsed	38.0	147.9	185.9	Elapsed	169.2	-27%	-9%
	305.7	415.8	721.5	CPU	136.4	482.6	619.0	CPU	545.8	-24%	-12%
	77.1	93.5	170.6	zIIP time	0.0	116.0	116.0	zIIP time	139.8	-18%	21%
			892.1	CPU+zIIP			735.0	CPU+zIIP	685.6	-23%	-7%
DB2SORT				Elapsed				Elapsed			
	198.6	123.2	321.8	CPU	165.4	268.8	434.2	CPU	348.8	8%	-20%
	15.9	370.7	386.6	zIIP time	123.3	218.0	341.3	zIIP time	282.8	-27%	-17%
	151.0	80.7	231.7	CPU+zIIP	0.0	204.9	204.9	CPU+zIIP	225.6	-3%	10%
		618.3				546.2		508.4	-18%	-7%	
30 input source datasets (one per partition)	91.4	122.9	214.3	Elapsed	33.9	142.7	176.6	Elapsed	160.2	-25%	-9%
	306.5	370.6	677.1	CPU	134.3	461.6	595.9	CPU	497.8	-26%	-16%
	76.7	80.9	157.6	zIIP time	0.0	105.9	105.9	zIIP time	113.8	-28%	7%
			834.7	CPU+zIIP			701.8	CPU+zIIP	611.6	-27%	-13%

Figure 9-1 LOAD PRESORT performance results

## Summary of results

When the results of scenario 3 are compared with scenario 1, the total elapsed time improvement is up to 27%, and an improvement of up to 23% in total CPU + zIIP time is observed for the combination of 30 input source data sets and **DFSORT**. An improvement of up to 8% in total CPU + zIIP time improvement is observed for single input source data set and **DFSORT** combined. The Db2 Sort improvement is either equivalent to **DFSORT** or better.

When the results of scenario 3 are compared with scenario 2, the total elapsed time improvement is up to 9%, and an improvement of up to 7% in total CPU + zIIP time is observed for 30 input source data sets and **DFSORT** combined. An improvement of up to 28% in total elapsed time improvement is observed when a single input source data set is used combined with **DFSORT**. The Db2 Sort improvement is either equivalent to **DFSORT** or better.

### 9.1.3 Usage considerations

Consider the following characteristics of the **LOAD PRESORT** option before you start using it:

- ▶ A clustering index is required. The clustering index can be implicit or it can be explicitly created.
- ▶ The sort of the input data can be done by either **DFSORT** or Db2 Sort.
- ▶ **PRESORT** requires **SORTDEVT** to dynamically allocate sort-related data sets.
- ▶ **PRESORT** is allowed for **SHRLEVEL NONE**, **REFERENCE**, and **CHANGE**.
- ▶ **PRESORT** is not the same as the existing **PRESORTED YES** keyword:
  - **PRESORT** means that the input records will be sorted in clustering order before the load.
  - **PRESORTED YES** means that the input data is in the correct order, so no sort is needed.
  - If both **PRESORT** and **PRESORTED YES** are specified, the **PRESORTED YES** option is ignored.

### 9.1.4 Conclusion

**LOAD PRESORT** helps to simplify the load process if you want to load tables in clustering key order when the source data is not in clustering key order. Also, you might benefit from reduced end-to-end elapsed times and a reduction of GP CPU time.

## 9.2 Online conversion from partition-by-growth to PBR

If you use a partition-by-growth (PBG) table space, as the table size grows, you might experience some drawbacks such as slower query performance, slower insert performance, or no inter-partition parallelism in utilities. If you experience these types of drawbacks, you might want to consider converting the table space from a PBG with  $m$  partitions to a PBR table space with  $n$  partitions ( $m$  and  $n$  are integer numbers here).

Before the availability of this feature, you had to unload the data, drop the PBG source table space, create a new PBR table space and a new target table, and load data into the new target PBR table space.

With this new Db2 13 feature, you can convert the table space from a PBG to a PBR table space (relative page numbering (RPN) only) by using a simple **ALTER TABLE ALTER PARTITIONING TO PARTITION BY RANGE** statement. If the data set of the PBG table space is not defined, the conversion is immediate. Otherwise, the **ALTER TABLE** statement puts the PBG table space in an advisory reorg (AREOR) pending state, and a subsequent **REORG TABLESPACE** with either **SHRLEVEL REFERENCE** or **CHANGE** is required to materialize the change.

### 9.2.1 Requirements

To use this feature, your Db2 environment must meet the following minimum requirements:

- ▶ Db2 13 at FL 500 or later
- ▶ APPLCOMPAT level V13R1M500 or later

## 9.2.2 Performance measurements

This section describes the measurements that were taken to assess the performance effects that are associated with enhancements to the online conversion from PBG to PBR process.

### Measurement environment

The performance evaluation was conducted on IBM z14 hardware with four general processors, one zIIP, and running z/OS 2.4 and Db2 13.

### Measurement scenario description

The PBG table in this test has approximately 180 million records with a record length of 142 bytes, and it uses three physical partitions (data sets). Two non-NPI indexes are used, and one of them is a clustering index.

Example 9-1 shows an **ALTER TABLE** statement that converts the partitioning scheme of table PBG\_TABLE from PBG to PBR. The target PBR table has three partitions with limit keys value\_1, value\_2, and MAXVALUE. The target PBR table's partition column is PARTITION\_COL.

*Example 9-1 ALTER TABLE alter partitioning to partition by range statement*

---

```
ALTER TABLE PBG_TABLE
  ALTER PARTITIONING TO PARTITION BY RANGE (PARTITION_COL)
  (PARTITION 1 ENDING AT ( value_1 ),
   PARTITION 2 ENDING AT ( value_2 ),
   PARTITION 3 ENDING AT ( MAXVALUE )
  ) ;
```

---

After the **ALTER TABLE** statement runs, the corresponding information is recorded in the SYSIBM.SYSPENDINGDDL catalog table, and the PBG table space is placed in the AREOR pending state. No other change occurs until the **REORG TABLESPACE** is done to materialize the change.

The following three scenarios were used to evaluate the performance improvements that are associated with enhancements to the online conversion from PBG to PBR process:

- ▶ Scenario 1 is a regular **REORG TABLESPACE** of the PBG table space with **SORTDATA YES** and **STATISTICS TABLE ALL INDEX ALL**. This scenario is the baseline for the comparison.
- ▶ Scenario 2 is a **REORG TABLESPACE** to convert the PBG table space to a 3-partition PBR table space. **SORTDATA YES** is enforced, and the default inline statistics with **TABLE ALL INDEX ALL** are collected.
- ▶ Scenario 3 is a **REORG TABLESPACE** to convert the PBG table space to a 30-partition PBR table space. **SORTDATA YES** is enforced, and the default inline statistics with **TABLE ALL INDEX ALL** are collected.

All three measurements were conducted with Db2 13. The elapsed time and the sum of CPU and zIIP time are used as metrics for the comparisons.

## Results

The measurement results are shown in Figure 9-2.

REORG phases	Scenario 1		Scenario 2				Scenario 3			
	REORG PBG 3 partitions		REORG to materialize PBG -> PBR with 3 partitions				REORG to materialize PBG -> PBR with 30 partitions			
	CPU+zIIP (s)	Elapsed (s)	CPU+zIIP	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1
UNLOAD	75.98	100.18	143.77	89%	169.01	69%	151.07	99%	177.64	77%
TS STATS SPHS*	102.70	294.56	101.32	-1%	295.00	0%	112.30	9%	334.55	14%
RELOAD	142.55	294.77	141.72	-1%	295.21	0%	116.47	-18%	335.48	14%
SORT STASKS	110.33	380.90	110.04	0%	381.57	0%	110.69	0%	421.29	11%
BUILD STASKS	57.97	380.91	57.02	-2%	381.58	0%	57.43	-1%	421.30	11%
IxStats	21.66	380.91	20.52	-5%	381.58	0%	21.28	-2%	421.30	11%
SORTBLD	0.16	86.36	0.17		86.59	0%	0.16		86.76	0%
LOG	0.00	0.08	0.24		0.38		0.08		0.84	
SWITCH	0.00	0.01	0.23		0.26		0.05		0.19	
INLINE COPY	10.17	381.19	9.85	-3%	381.86	0%	32.11	216%	422.48	11%
TOTAL	521.52	481.81	584.88	12%	551.59	14%	601.63	15%	601.49	25%

Figure 9-2 Online conversion of PBG to PBR performance results

TS STATS SPHS is the abbreviation of *table space statistics subphase*.

### Summary of results

The following list summarizes the key points of this performance assessment:

- ▶ The materializing **REORG** scenarios (scenarios 2 and 3) have a higher cost in the **UNLOAD** phase because **REORG** must extract the limit key and evaluate the new partition number in the target PBR table space for each record that is unloaded. Another contributing factor is the building the compression dictionaries: Instead of one dictionary, a separate compression dictionary is built for each PBR partition.
- ▶ The materializing **REORG** has a higher cost for collecting table statistics and performing the **INLINE COPY** when the number of PBR partitions is higher.
- ▶ Although the materializing **REORG** has a higher value that is recorded for the **LOG** and **SWITCH** phases, the number remains less than 1 second in the test.

### 9.2.3 Usage considerations

The following considerations apply when converting from a PBG to a PBR table space:

- ▶ The NPI indexes of the original PBG table remain NPIs after conversion to PBR. The conversion does not create a PI, and it does not convert any NPIs to PIs.
- ▶ The target PBR table space must use RPN. Conversion to a PBR absolute page numbering (APN) is not supported.
- ▶ The **REORG TABLESPACE** to materialize the conversion can use either **SHRLEVEL REFERENCE** or **SHRLEVEL CHANGE**.

## 9.2.4 Conclusion

The Db2 13 online conversion from PBG to PBR capabilities provides a simple and convenient way to convert a PBG table space to a PBR table space. Although the **REORG** to materialize the conversion does result in some overhead when compared to a regular **REORG** of the PBG table space when the target PBR table has many partitions, you likely will experience a similar (if not larger) amount of overhead, and an outage if you use the traditional conversion method (unload and drop the PBG, create the PBR, and then reload your data).

## 9.3 Page sampling support for inline statistics

Db2 10 introduced page sampling for the **RUNSTATS** utility. With page sampling, only a percentage of pages from the table is passed to the statistics subtask to collect statistics, and that percentage is determined by the sampling rate. For example, with a sampling rate of 10, only 10% of the table's pages are sampled for statistics. **RUNSTATS** with page sampling can greatly reduce elapsed time and CPU time when distribution and column statistics are collected on a large table.

However, page sampling was not supported when using inline statistics. This Db2 13 feature fills this gap by adding page sampling support for **REORG TABLESPACE** and **LOAD REPLACE** by gathering inline statistics for cardinalities, frequencies, distribution, and histogram statistics. To invoke the page sampling method, the keyword **TABLESAMPLE SYSTEM** is added to the **REORG** and **LOAD** syntax with the following options:

<b>AUTO</b>	Db2 determines the sampling rate based on the size of the table: the larger the table, the smaller the sampling rate. There is an internal threshold of 500,000. If the table has fewer rows than the threshold, page sampling is not used, and all the table's pages go through the statistics collection process.
<b>NONE</b>	No page sampling occurs.
<b>A numeric-literal</b>	A user-provided positive number 1 - 100 that is used as the sampling rate.

The **SAMPLE** keyword existed for sampling inline statistics before this enhancement. This keyword is intended for row-level sampling in which all the records are passed to the statistics subtask, and the statistics subtask selects a portion of the records for statistics collection. Page sampling should result in a greater cost reduction than row-level sampling because fewer rows are passed to the statistics subtask and fewer rows must be sorted, if sorting is required for distribution or column statistics.

Similar to the **RUNSTATS** utility, the existing **STATGSAMP** subsystem parameter determines whether to use page sampling by default for inline statistics. **STATGSAMP** provides the following options:

<b>SYSTEM (default) or YES</b>	<b>RUNSTATS</b> or inline statistics always run as though <b>TABLESAMPLE SYSTEM AUTO</b> is specified, unless <b>TABLESAMPLE SYSTEM NONE</b> or a numeric-literal value is specified in the statement. Even when the <b>SAMPLE</b> keyword is specified in the inline statistics statement, row-level sampling is ignored, and page sampling is used instead.
<b>NO</b>	<b>RUNSTATS</b> or inline statistics do not use page sampling by default. Any sampling is determined by what is specified in the statement.

## 9.3.1 Requirements

To use this feature, you must be using Db2 13 at FL 500 or later.

## 9.3.2 Performance measurements

This section describes the measurements that were taken to assess the performance effects that are associated with enhancements to inline statistics page sampling.

### Measurement environment

The performance evaluation was conducted on IBM z14 hardware with four general processors, one zIIP, and running z/OS 2.4 and Db2 13.

### Measurement scenario description

The following four scenarios were used to evaluate the performance improvements that are associated with enhancements to inline statistics page sampling:

- ▶ Scenario 1 uses **REORG** or **LOAD** with **STATISTICS** without sampling. The measurements were taken before the integration of this feature into the Db2 13 code base.
- ▶ Scenario 2 uses **REORG** or **LOAD** with **STATISTICS TABLESAMPLE SYSTEM NONE**. The measurements verify whether **TABLESAMPLE SYSTEM NONE** has similar performance as scenario 1.
- ▶ Scenario 3 uses **REORG** or **LOAD** with **STATISTICS TABLESAMPLE SYSTEM AUTO**, which uses page sampling. The sampling rate is 10 in this scenario, which is determined by the Db2 utility based on the size of the table. The measurements are designed to compare how much page sampling outperforms no sampling (scenario 1) and row-level sampling (scenario 4).
- ▶ Scenario 4 uses **REORG** or **LOAD** with row-level sampling.

In each scenario, the test is performed on a PBR table space with 30 logical partitions (LPARs) and a PBG table space with three physical partitions. Both the PBR and the PBG table space contain approximately 180 million records with a record length 142 bytes. There is no index on either of the tables because page sampling applies only to the table space, not the index.

For each scenario, the tests collect two types of statistics: the first collects simple table statistics, and the other collects more complex statistics (table statistics plus 10 frequencies and 10 histogram quantiles for each of the seven column groups).

### Results

The results of this evaluation are presented in two parts. The measurements for **REORG TABLESPACE STATISTICS** with page-level sampling are shown first, followed by the same measurements for **LOAD REPLACE STATISTICS** with page-level sampling.

#### ***Performance of REORG TABLESPACE STATISTICS with page-level sampling***

Figure 9-3 on page 287 shows that **TABLESAMPLE SYSTEM NONE** has equivalent performance to no sampling (before the integration of this feature) on a PBR table space. Using **TABLESAMPLE SYSTEM AUTO** reduces the CPU + zIIP time by 89 - 90% for the statistics collection processing compared to no sampling. The result is a 26% reduction in CPU + zIIP time and an 8% reduction in elapsed time for the overall **REORG** job when the “simple” statistics are being collected, a 73% reduction in CPU + zIIP time, and a 75% reduction in elapsed time for the overall **REORG** process when “complex” statistics are gathered.

PBR description	Utility	Scenario 1 Db2 13 REORG inline statistics no sample		Scenario 2 Db2 13 REORG inline statistics TABLESAMPLE SYSTEM NONE				Scenario 3 Db2 13 REORG inline statistics TABLESAMPLE SYSTEM AUTO			
		Phases	CPU+zIIP (s)	Elapsed (s)	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)
Table is PBR with 30 partitions, 180 million rows, record length 142 bytes, no index. REORG TABLESPACE SHRLEVEL CHANGE STATISTICS TABLE ALL	UNLOAD	0.49	29.86	0.49	-1%	28.91	-3%	0.50	2%	28.76	-4%
	Reload SPHS	218.08	104.70	212.03	-3%	103.37	-1%	220.42	1%	88.18	-16%
	TS STATS SPHS	134.62	75.25	140.30	4%	74.96	0%	14.33	-89%	59.84	-20%
	RELOAD	0.08	77.60	0.08	5%	77.25	0%	0.08	5%	62.10	-20%
	Sort stasks	51.03	159.20	50.73	-1%	158.97	0%	50.74	-1%	144.49	-9%
	Build stasks	28.28	159.22	28.29	0%	158.99	0%	28.86	2%	144.51	-9%
	SORTBLD	0.03	83.97	0.03	-14%	84.04	0%	0.03	-6%	84.67	1%
	INLINE COPY	13.13	161.90	13.23	1%	161.59	0%	13.14	0%	147.04	-9%
	TOTAL	445.74	193.80	445.17	0%	192.53	-1%	328.09	-26%	177.74	-8%
		Phases	CPU+zIIP (s)	Elapsed (s)	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)
Table is PBR with 30 partitions, 180 million rows, record length 142 bytes, no index. REORG TABLESPACE SHRLEVEL CHANGE STATISTICS TABLE ALL, COLGRUPS FREQUENCYs and HISTOGRAMs	UNLOAD	0.58	30.57	0.57	-1%	30.52	0%	0.59	2%	30.55	0%
	Reload SPHS	223.10	198.44	220.92	-1%	195.84	-1%	219.05	-2%	95.41	-52%
	TS STATS SPHS	880.13	880.08	867.48	-1%	866.51	-2%	84.39	-90%	133.53	-85%
	DSTATSRT SPHS	485.51	880.12	482.36	-1%	866.56	-2%	50.88	-90%	133.54	-85%
	RELOAD	0.10	862.46	0.10	2%	868.93	-2%	0.10	3%	135.94	-85%
	Sort stasks	57.19	963.49	56.41	-1%	952.21	-1%	51.87	-9%	217.59	-77%
	Build stasks	28.08	963.51	29.36	5%	952.23	-1%	28.50	1%	217.61	-77%
	SORTBLD	0.02	83.31	0.02	-11%	85.60	3%	0.01	-25%	83.99	1%
	INLINE COPY	14.91	966.10	14.68	-2%	954.86	-1%	13.69	-8%	220.19	-77%
	TOTAL	1689.63	999.33	1671.91	-1%	987.98	-1%	449.10	-73%	253.10	-75%

Figure 9-3 Performance results for page-level sampling of inline statistics with REORG on a PBR table space

DSTATSRT is the abbreviation of *distribution statistics sort*.

Figure 9-4 shows the REORG measurements for a PBG table space with inline statistics that uses page-level sampling. These results are similar to the performance improvements when using a PBR table space.

PBG description	Utility	Scenario 1 Db2 13 REORG inline statistics no sample		Scenario 2 Db2 13 REORG inline statistics TABLESAMPLE SYSTEM NONE				Scenario 3 Db2 13 REORG inline statistics TABLESAMPLE SYSTEM AUTO				
		Phases	CPU+zIIP (s)	Elapsed (s)	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1
Table is PBG with 3 physical partitions, 180 million rows, record length 142 bytes, no index. REORG TABLESPACE SHRLEVEL CHANGE STATISTICS TABLE ALL	UNLOAD	32.88	85.87	32.80	0%	88.82	3%	32.84	0%	89.64	4%	
	TS STATS SPHS	127.96	278.34	133.30	4%	282.33	1%	13.48	-89%	178.86	-36%	
	RELOAD	102.83	278.44	103.07	0%	282.43	1%	97.70	-5%	178.96	-36%	
	Sort	0.00	0.03	0.00	-8%	0.03	3%	0.00	-7%	0.04	13%	
	Build	14.22	77.43	14.16	0%	77.68	0%	14.15	-1%	77.66	0%	
	Parallel sortkeys	18.99	355.64	19.15	1%	359.88	1%	19.03	0%	256.35	-28%	
	INLINE COPY	9.78	355.99	9.67	-1%	360.22	1%	9.57	-2%	256.72	-28%	
	TOTAL	306.66	442.41	312.15	2%	449.56	2%	186.75	-39%	346.75	-22%	
		Phases	CPU+zIIP (s)	Elapsed (s)	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1
	Table is PBG with 3 physical partitions, 180 million rows, record length 142 bytes, no index. REORG TABLESPACE SHRLEVEL CHANGE STATISTICS TABLE ALL, COLGRUPS FREQUENCYs and HISTOGRAMs	UNLOAD	32.72	86.60	32.66	0%	88.98	3%	32.76	0%	87.50	1%
TS STATS SPHS		855.54	1175.10	855.65	0%	1173.96	0%	85.99	-90%	258.35	-78%	
DSTATSRT SPHS		443.83	1175.44	452.10	2%	1174.33	0%	42.45	-90%	258.37	-78%	
RELOAD		102.09	1175.62	101.40	-1%	1174.50	0%	98.06	-4%	258.58	-78%	
Sort		0.00	0.04	0.00	0%	0.04	0%	0.00	-23%	0.04	0%	
Build		14.25	77.41	14.15	-1%	77.27	0%	14.14	-1%	77.16	0%	
Parallel sortkeys		19.65	1252.75	19.86	1%	1251.49	0%	18.85	-4%	335.43	-73%	
INLINE COPY		10.59	1253.15	10.71	1%	1251.91	0%	9.95	-6%	335.85	-73%	
TOTAL		1478.65	1340.37	1486.53	1%	1341.40	0%	302.20	-80%	423.86	-68%	

Figure 9-4 Performance results for page-level sampling of inline statistics with REORG on a PBG table

Figure 9-5 shows that using page-level sampling reduces CPU + zIIP time in the statistics collection process by 10% when compared to row-level sampling for “simple” statistics. It also shows an 89 - 91% CPU + zIIP time reduction when page-level sampling is used to collect “complex” statistics instead of using row-level sampling.

PBG description STATPGSAMP=NO	Utility	Scenario 4 Db2 13 REORG inline statistics sample 10		Scenario 3 Db2 13 REORG inline statistics TABLESAMPLE SYSTEM AUTO			
		Phases	CPU+zIIP (s)	Elapsed (s)	CPU+zIIP (s)	% diff vs. Scenario 4	Elapsed (s)
Table is PBG with 3 physical partitions, 180 million rows, record length 142 bytes, no index. REORG TABLESPACE SHRLEVEL CHANGE STATISTICS TABLE ALL	UNLOAD	32.63	84.02	32.84	1%	89.64	7%
	TS STATS SPHS	15.01	186.77	13.48	-10%	178.86	-4%
	RELOAD	99.82	186.88	97.70	-2%	178.96	-4%
	SORT	0.00	0.04	0.00	-1%	0.04	-10%
	BUILD	13.75	77.16	14.15	3%	77.66	1%
	Parallel sortkeys	19.51	263.81	19.03	-2%	256.35	-3%
	INLINE COPY	9.82	264.16	9.57	-3%	256.72	-3%
	TOTAL	190.55	348.78	186.75	-2%	346.75	-1%
Table is PBG with 3 physical partitions, 180 million rows, record length 142 bytes, no index. REORG TABLESPACE SHRLEVEL CHANGE STATISTICS TABLE ALL, COLGRUPs FREQUENCYs and HISTOGRAMs	Phases	CPU+zIIP (s)	Elapsed (s)	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1
	UNLOAD	32.29	83.40	32.76	1%	87.50	5%
	TS STATS SPHS	753.39	1158.91	85.99	-89%	258.35	-78%
	DSTATSRT SPHS	466.77	1159.27	42.45	-91%	258.37	-78%
	RELOAD	100.61	1159.46	98.06	-3%	258.58	-78%
	SORT	0.00	0.04	0.00	-30%	0.04	-5%
	BUILD	13.75	77.49	14.14	3%	77.16	0%
	Parallel sortkeys	21.32	1236.66	18.85	-12%	335.43	-73%
INLINE COPY	11.14	1237.08	9.95	-11%	335.85	-73%	
TOTAL	1399.26	1320.96	302.20	-78%	423.86	-68%	

Figure 9-5 Page-level versus row-level sampling with REORG on a PBG table space

**Performance of LOAD REPLACE STATISTICS with page-level sampling**

Figure 9-6 shows the performance results for LOAD REPLACE that uses inline statistics for a PBR table space when using page-level sampling versus no sampling.

PBR description NUMRECS provided	Utility	Scenario 1 Db2 13 LOAD inline statistics no sample		Scenario 2 Db2 13 LOAD inline statistics TABLESAMPLE SYSTEM NONE			Scenario 3 Db2 13 LOAD inline statistics TABLESAMPLE SYSTEM AUTO				
		Phases	CPU+zIIP (s)	Elapsed (s)	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)
Table is PBR with 30 partitions, 180 million rows, record length 142 bytes, no index. Single input source dataset	TS STATS SPHS	112.84	309.47	109.68	-3%	311.91	1%	10.78	-90%	238.13	-23%
	RELOAD	194.23	312.03	196.27	1%	314.36	1%	201.47	4%	240.50	-23%
	TOTAL	307.07	312.85	305.95	0%	315.16	1%	212.24	-31%	241.10	-23%
Table is PBR with 30 partitions, 180 million rows, record length 142 bytes, no index. Single input source dataset LOAD STATISTICS TABLE ALL, COLGRUPs FREQUENCYs and HISTOGRAMs	Phases	CPU+zIIP (s)	Elapsed (s)	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1
	TS STATS SPHS	402.43	756.53	382.38	-5%	748.85	-1%	37.93	-91%	278.15	-63%
	DSTATSRT SPHS	433.58	756.75	431.58	0%	749.08	-1%	40.38	-91%	278.18	-63%
	RELOAD	196.58	759.18	199.57	2%	751.51	-1%	203.93	4%	280.58	-63%
TOTAL	1032.59	760.21	1013.53	-2%	752.58	-1%	282.25	-73%	281.62	-63%	
Table is PBR with 30 partitions, 180 million rows, record length 142 bytes, no index. Partition level input source datasets LOAD STATISTICS TABLE ALL	Phases	CPU+zIIP (s)	Elapsed (s)	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1
	LD STACKS	286.38	90.97	288.66	1%	91.05	0%	314.53	10%	75.81	-17%
	TS STATS SPHS	117.07	91.02	112.92	-4%	91.11	0%	11.41	-90%	75.88	-17%
	RELOAD	0.09	93.50	0.09	-6%	93.56	0%	0.09	0%	78.37	-16%
TOTAL	403.55	94.09	401.66	0%	94.16	0%	326.04	-19%	78.98	-16%	
Table is PBR with 30 partitions, 180 million rows, record length 142 bytes, no index. Partition level input source datasets LOAD STATISTICS TABLE ALL, COLGRUPs FREQUENCYs and HISTOGRAMs	Phases	CPU+zIIP (s)	Elapsed (s)	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1
	LD STACKS	265.21	195.84	268.89	1%	193.78	-1%	308.93	16%	91.00	-54%
	TS STATS SPHS	303.79	339.16	379.48	25%	325.13	-4%	38.18	-87%	103.84	-69%
	DSTATSRT SPHS	489.56	450.16	480.76	-2%	440.80	-2%	47.43	-90%	115.41	-74%
	RELOAD	0.69	454.17	0.71	2%	444.76	-2%	0.70	1%	119.43	-74%
TOTAL	1059.25	455.16	1129.84	7%	445.75	-2%	395.24	-63%	120.44	-74%	

Figure 9-6 Performance results for page-level sampling of inline statistics with LOAD REPLACE on a PBR table space



Figure 9-7 shows the results of a similar set of measurements, but here a PBR table space is used instead of a PBG table space.

PBG description NUMRECS provided	Utility	Scenario 1 Db2 13 LOAD inline statistics no sample		Scenario 2 Db2 13 LOAD inline statistics TABLESAMPLE SYSTEM NONE				Scenario 3 Db2 13 LOAD inline statistics TABLESAMPLE SYSTEM AUTO			
		Phases	CPU+zIIP (s)	Elapsed (s)	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)
Table is PBG with 30 partitions, 180 million rows, record length 142 bytes, no index. Single input source dataset	TS STATS SPHS	102.22	300.24	100.98	-1%	297.46	-1%	10.18	-90%	212.51	-29%
	RELOAD	192.63	300.58	190.63	-1%	297.80	-1%	184.39	-4%	212.85	-29%
	TOTAL	294.85	300.65	291.62	-1%	297.87	-1%	194.57	-34%	212.92	-29%
Table is PBG with 30 partitions, 180 million rows, record length 142 bytes, no index. Single input source dataset LOAD STATISTICS TABLE ALL, COLGRUPS FREQUENCYs and HISTOGRAMs	Phases	CPU+zIIP (s)	Elapsed (s)	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1	CPU+zIIP (s)	% diff vs. Scenario 1	Elapsed (s)	% diff vs. Scenario 1
	TS STATS SPHS	371.76	730.77	361.02	-3%	719.37	-2%	37.52	-90%	252.04	-66%
	DSTATSRT SPHS	444.74	731.00	442.22	-1%	719.62	-2%	41.76	-91%	252.07	-66%
	RELOAD	190.97	731.37	190.94	0%	719.98	-2%	185.15	-3%	252.43	-65%
TOTAL	1007.47	731.50	994.18	-1%	720.12	-2%	264.44	-74%	252.56	-65%	

Figure 9-7 Performance results for page-level sampling of inline statistics with LOAD REPLACE on a PBR table space

As shown in Figure 9-6 on page 288 and Figure 9-7, using inline statistics with page-level sampling with the **LOAD** utility produces performance improvements that are similar to the ones for the **REORG** utility.

### Summary of results

Using page sampling with a sampling rate of 10 at the inline statistics phase in **LOAD** and **REORG** saves approximately 90% of CPU + zIIP time compared to no sampling, regardless of whether “simple” statistics or “complex” statistics are collected.

When the inline statistics collection phase uses a page-level sampling rate of 10, CPU + zIIP time for “simple” statistics is reduced by approximately 10%, and CPU + zIIP time for “complex” statistics is reduced by approximately 90% when compared to using row-level sampling.

Because the cost of collecting complex statistics, such as distribution or column group statistics, weighs more in the overall **REORG** or **LOAD** job, the overall CPU and elapsed time savings from using page-level sampling for the entire job is more significant.

### 9.3.3 Usage considerations

Consider the following characteristics of page-level sampling before you start using it:

- ▶ Message DSNU1374I indicates the actual sample rate that is used by utility.
- ▶ You should not expect the statistics that are collected through page-level sampling to be the same each time that you run the utility because sampling introduces some degree of randomness. However, the results should be in the same general range.
- ▶ Page sampling saves cost and time, but the statistics that are collected are not as accurate as when sampling is not used, especially if the data is skewed. For example, in determining the filter factor of predicate C1 IN (1,3,5), if C1 has a high frequency of value 1, but the frequency is not as accurate due to page sampling, the filter factor might not be calculated as precisely.

**Important:** If the existence or accuracy of the statistics is crucial to your access path selection, consider the following options:

- ▶ Use **TABLESAMPLE SYSTEM numeric-literal** to specify a higher sample rate.
- ▶ Specify **TABLESAMPLE SYSTEM NONE** in the inline statistics statement.
- ▶ Set the **STATGSAMP** subsystem parameter to **N0** to disable page sampling.

- ▶ A best practice is when you use page-level sampling with **LOAD** is to specify the **NUMRECS** keyword in the statement to offer an accurate number of rows to get a better estimation of the required page sampling rate.

### 9.3.4 Conclusion

Because inline statistics with page-level sampling can result in reductions in CPU and elapsed time compared to no sampling or row-level sampling, it is a best practice to use page-level sampling whenever possible with existing inline statistics jobs. When the **STATGSAMP** subsystem parameter is set to **SYSTEM** (default), page-level sampling is enabled automatically. If you do not want to use page-level sampling, consider setting **STATPGSAMP** to **NO** or specifying **TABLESAMPLE SYSTEM NONE** in the inline statistics statements to disable it.

## 9.4 Utility history

As a DBA or system administrator running Db2 utilities to, for example, load tables, reorganize table spaces and indexes, rebuild indexes, recover objects, or collect statistics, you want to track historical information that is related to the utilities. This information includes but is not limited to start and end time; elapsed and CPU time; whether the job was successful; and whether it ran longer or cost more CPU compared to previous runs. With a new Db2 13 feature, you can track this type of historical information by using new Db2 catalog tables.

You can enable this feature by setting the **UTILITY\_HISTORY** subsystem parameter to **UTILITY** (**NONE** is the default value) directly in the **DSNTIJUZ** installation job, or you can specify it on the **DSNTIP63** installation panel (Db2 utilities parameters panel 4). This feature is available in **FL V13R1M501** and later.

The utilities history records are stored in the new **SYSIBM.SYSUTILITIES** catalog table. For each utility run, a row is inserted and updated in this table to record the job name, utility name, starting and ending timestamp, elapsed time, CPU time, zIIP time, and return code. You can query this table to retrieve information about a certain utility job or to do some analysis about past runs of certain utilities.

You can collect history records for the following Db2 utilities:

- ▶ **BACKUP SYSTEM**
- ▶ **CATMAINT**
- ▶ **CHECK DATA**
- ▶ **CHECK INDEX**
- ▶ **CHECK LOB**
- ▶ **COPY**
- ▶ **COPYTOCOPY**
- ▶ **LOAD**
- ▶ **MERGECOPY**
- ▶ **MODIFY RECOVERY**
- ▶ **MODIFY STATISTICS**
- ▶ **QUIESCE**
- ▶ **REBUILD IDNEX**
- ▶ **RECOVER**
- ▶ **REORG**
- ▶ **REPAIR**
- ▶ **RUNSTATS**
- ▶ **UNLOAD**

- ▶ **REPORT RECOVERY**
- ▶ **REPORT TABLESPACESET**
- ▶ **STOSPACE**

## 9.4.1 Requirements

To use this feature, you must be using Db2 13 at FL 501 or later.

## 9.4.2 Performance measurements

This section describes the measurements that were taken to assess the performance of the utility history feature.

### Measurement environment

The performance evaluation was conducted on IBM z14 hardware with four general processors, one zIIP, and running z/OS 2.4 and Db2 13.

### Measurement scenario description

This test uses a table in a PBR table space with 30 partitions. The table contains approximately 180 million rows with a record length of 142 bytes. The table has two indexes.

The performance test of this feature includes the following seven utilities:

- ▶ **LOAD**
- ▶ **REBUILD INDEX**
- ▶ **COPY**
- ▶ **REORG**
- ▶ **RUNSTATS**
- ▶ **RECOVER**
- ▶ **UNLOAD**

Because only one record is inserted and updated in the SYSIBM.SYSUTILITIES table for each utility job, you can expect a slight amount of overhead when utility history information is collected. The main purpose of this test is to ensure that the performance-related information (such as elapsed time, CPU time, and zIIP time) in SYSIBM.SYSUTILITIES matches the numbers in accounting reports.

## Results

The results of the test are shown in Figure 9-8. No obvious overhead is observed when utility history information is collected (shown in the third group of columns from the left) compared to not collecting utility history information (shown in the second group of columns from the left).

The numbers from SYSIBM.SYSUTILITIES (the first group of columns from the right) match the numbers from the Db2 accounting reports (second group of columns from the right).

	Db2 13, from accounting reports, no utility history			Db2 13, from accounting reports, collect utility history			Db2 13, from SYSUTILITIES, collect utility history		
	CPU	zIIP	Elapsed time	CPU	zIIP	Elapsed time	CPU	zIIP	Elapsed time
LOAD (DB2 SORT)	393.88	99.31	149.80	407.22	98.35	141.18	407.77	103.80	141.17
LOAD (DFSORT)	421.51	103.95	149.89	433.00	107.85	154.27	432.97	107.85	154.25
RBLDX-1	122.59	17.96	46.73	120.06	18.11	46.01	120.05	18.11	46.00
RBLDX-2	119.31	16.49	44.61	119.97	16.56	44.87	119.97	16.56	44.86
COPY	12.03	0.00	7.47	12.19	0.00	7.50	12.18	0.00	7.50
REORG	327.39	83.53	181.17	328.25	82.92	184.78	328.22	82.92	184.76
RUNST	11.16	1.38	13.18	11.05	1.38	13.10	11.05	1.38	13.09
RECVR	7.00	0.00	11.37	7.20	0.00	11.43	7.18	0.00	11.41
UNLDP	267.00	0.00	83.22	275.20	0.00	83.77	275.20	0.00	83.77

Figure 9-8 Performance results for utility history collection

### 9.4.3 Usage considerations

The size of the SYSIBM.SYSUTILITIES table grows as more records are inserted by each utility job. Db2 does not provide a cleanup process to remove old records. Therefore, as a best practice, periodically delete records that are no longer needed.

### 9.4.4 Conclusion

DBAs or system administrators who run Db2 utilities find that the utility history feature is a convenient tool that provides relief from the burden of manually managing the bookkeeping of past utility runs.

## 9.5 Redirected recovery

Most Db2 system administrators probably have experienced some pressure when performing a recovery of Db2 objects. Whether the service-level agreement (SLA) recovery time objectives (RTO) can be met or how risky the recovery will be influences the decision about which recovery strategy to use. Knowing the (expected) duration of the recovery is critical, but this information is not easy to obtain. Testing on the production system impacts the applications, and testing on a test system requires the same configuration as production. Replicating and manipulating the production system's image copies, Db2 logs, and bootstrap data sets (BSDSs) so that they can be used on the test system is a time-consuming and complex process.

To help alleviate the difficulties that are associated with recovering database objects, Db2 added a redirected recovery option for table spaces and index spaces to the Db2 **RECOVER** utility. To leverage this feature, you must have a target table space and index space with the same definition as the definition of the source, and both must be in the same Db2 subsystem.

Redirected recovery uses the source object's image copies and Db2 log records to recover into the target object and converts the object identifiers (OBIDs) in the target table space and index space at the end. By using redirected recovery, you do not need to test a recovery on a test Db2 subsystem or prepare the recovery resources; the recovery does not cause outages of the source table space or index space; and applications are not impacted. A redirected recovery that uses the target object should have similar performance as a recovery that uses the actual (production) source object.

Redirected recovery can recover the target object to the current state or to a point in time. If an error occurs during the recovery (such as a missing image copy), it is revealed during the redirected recovery process. If the elapsed time of the redirected recovery does not meet your RTO, certain actions can be taken to improve recovery performance, such as taking more frequent image copies to make sure that the RTO can be met if actual recovery is needed.

## 9.5.1 Requirements

To use this feature, your Db2 environment must meet the following minimum requirements:

- ▶ Db2 12 at FL 500 or later or Db2 13
- ▶ For Db2 12, APAR PH27043 for table spaces and APAR PH32566 for index spaces

## 9.5.2 Performance measurements

This section describes the measurements that were taken to assess the performance effects of the redirected recovery feature.

### Measurement environment

The performance evaluation was conducted on IBM z14 hardware with four processors, no zIIPs, and running z/OS 2.3 and a Db2 12 non-data-sharing system.

Buffer pool 0 (BP0) is used for the Db2 catalog objects. Both the source and target table spaces and index spaces are assigned to the 4 K buffer pools with VPSIZE 1,000,000. The Db2 log buffer (OUTBUF) is defined as 30,000 buffers.

### Measurement scenarios and results

The source and target objects are identical in terms of table space, table, and index definition. The table spaces that are used for the tests are PBR with 100 partitions and 100 million records. Two indexes are defined: 1 PI and 1 NPI.

After the image copy is taken for the source object, 10 million records are updated in the source table.

The scenarios that were used to evaluate redirected recovery and their results are presented in two parts. Information about redirected recovery of the target table space is shown first, followed by information about redirected recovery of indexes.

### Scenarios and results for redirected recovery of the target table space

The following four scenarios were used to evaluate the performance improvements that are associated with redirected recovery of the target table space:

- ▶ Scenario 1: Recover the table space to the current state.
  - Run **RECOVER TABLESPACE** on the source table space by using the Db2 12 base code level.
  - Run **RECOVER TABLESPACE** on the source table space by using a code level that supports redirected recovery.
  - Run **RECOVER TABLESPACE** from the source table space to the redirected target table space by using a code level that supports redirected recovery.

As shown in Figure 9-9, no performance degradation is observed when the results of running **RECOVER** with a code level that supports redirected recovery are compared to the results of running **RECOVER** with the base code level.

Comparing the results of recovering the actual source table space (by using the base code level) against the results of a redirected recovery of the target table space from the source table space shows up to a 6% variation in total CPU time and up to a 2% variation in total elapsed time.

Table space of 100 millions rows and 100 partitions, 2 indexes 10 million rows were updated	Recover source table space						Redirect recover target table space				
	Db2 12 base		With Db2 12 APARs		delta %Diff (base vs APARs)		With Db2 12 APARs		delta %Diff (base vs APARs)		
	CPU	Elapsed time	CPU	Elapsed time	CPU	Elapsed time	CPU	Elapsed time	CPU	Elapsed time	
RECOVER single table space to current LOGAPPLY PHASE: • 20,219,372 LOG RECORDS READ • 10,294,118 LOG RECORDS APPLIED											
<b>Total Job</b>	17.37	87.96	17.57	87.69	1%	0%	16.49	89.07	-6%	2%	
Restore Read	2.710	74.24	2.637	73.99	-3%	0%	2.701	76.18	2%	3%	
Restore Write	3.303	74.24	3.395	73.99	3%	0%	3.295	76.18	-3%	3%	
Log Apply	10.47	12.26	10.65	12.33	2%	1%	9.61	11.42	-10%	-7%	
Translat							0.001	0.103			

Figure 9-9 Performance of RECOVER TABLESPACE on a source versus a redirected target

- ▶ Scenario 2: Recover the table space to a point-in-time by using the **TORBA** recovery option.
  - Run **RECOVER TABLESPACE TORBA** on the source table space by using the Db2 12 base code level.
  - Run **RECOVER TABLESPACE TORBA** on the source table space by using a code level that supports redirected recovery.
  - Run **RECOVER TABLESPACE TORBA** from the source table space to the redirected target table space by using a code level that supports redirected recovery.

Figure 9-10 on page 295 shows that no performance degradation is observed when the results of recovering the source object by using **RECOVER TORBA** with the Db2 base code level are compared against the results of recovering the source object by using a code level that supports redirected recovery.

Comparing the results of recovering the source table by using **RECOVER TORBA** with the base code level against the results of recovering the source table space to the target table space by using redirected **RECOVER TORBA** from the source table space to the target table space shows up to a 6% variation in overall CPU time and up to a 10% variation in overall elapsed time. There is an 18% CPU and 29% elapsed time difference in the LOGUNDO phase, which is because redirected recovery does not write compensation log records during the LOGUNDO phase.

Table space of 100 millions rows and 100 partitions 2 indexes 10 million rows were updated	Recover source table space						Redirect recover target table space			
	Db2 12 base		With Db2 12 APARs		delta %Diff (base vs APARs)		With Db2 12 APARs		delta %Diff (base vs APARs)	
	CPU	Elapsed time	CPU	Elapsed time	CPU	Elapsed time	CPU	Elapsed time	CPU	Elapsed time
RECOVER single Tablespace TORBA LOGAPPLY PHASE: • 10,241,933 LOG RECORDS READ • 10,241,911 LOG RECORDS APPLIED LOGCSR PHASE: • 308,983 LOG RECORDS READ • 0 LOG RECORDS APPLIED LOGUNDO PHASE: • 10,241,916 LOG RECORDS READ • 9,949,284 LOG RECORDS APPLIED										
<b>Total Job</b>	28.99	135.78	28.98	136.85	0%	1%	27.19	123.38	-6%	-10%
Restore Read	2.723	72.66	2.695	73.19	-1%	1%	2.579	74.48	-4%	2%
Restore Write	3.391	72.66	3.287	73.19	-3%	1%	3.290	74.48	0%	2%
Log Apply	8.502	8.662	8.659	8.905	2%	3%	9.304	9.473	7%	6%
LogCSR	0.133	0.135	0.136	0.138	2%	2%	0.136	0.138	0%	0%
LogUNDO	13.36	52.97	13.33	53.27	0%	1%	10.99	37.80	-18%	-29%
Translat							0.001	0.103		

Figure 9-10 Performance of RECOVER TABLESPACE TORBA on a source versus redirected target

- ▶ Scenario 3: Recover a table space list of 100 partitions to the current state.
  - Run a **RECOVER TABLESPACE** list of 100 partitions on the source table space by using the Db2 12 base code level.
  - Run a **RECOVER TABLESPACE** list of 100 partitions on the source table space by using a code level that supports redirected recovery.
  - Run a **RECOVER TABLESPACE** list of 100 partitions from the source table space to the redirected target table space by using a code level that supports redirected recovery.

Figure 9-11 shows that no performance degradation is observed when the results of running **RECOVER** for a list of 100 partitions at the base code level are compared against the results of using a code level that supports redirected recovery.

Figure 9-11 also shows that recovering a list of 100 partitions on the source table space (by using the Db2 base code level), and redirected recovery (by using the code level that supports redirected recovery) from the source table space to the target table space of a list of 100 partitions has a similar overall CPU time and elapsed time.

Table space of 100 millions rows and 100 partitions, 2 indexes 10 million rows were updated	Recover source table space list of 100 partitions						Redirect recover target table space list of 100 partitions			
	Db2 12 base		With Db2 12 APARs		delta %Diff (base vs APARs)		With Db2 12 APARs		delta %Diff (base vs APARs)	
	CPU	Elapsed time	CPU	Elapsed time	CPU	Elapse d time	CPU	Elapsed time	CPU	Elapsed time
RECOVER LIST OF 100 PARTITIONS LOGAPPLY PHASE: • 25,883,227 LOG RECORDS READ • 12,941,200 LOG RECORDS APPLIED										
<b>TOTAL Job</b>	<b>21.99</b>	<b>114.84</b>	<b>21.37</b>	<b>112.12</b>	<b>-3%</b>	<b>-2%</b>	<b>22.09</b>	<b>113.61</b>	<b>3%</b>	<b>1%</b>
Restore Read	2.117	72.51	1.940	71.52	-8%	-1%	2.166	71.75	12%	0%
Restore Write	3.036	72.51	3.059	71.52	1%	-1%	3.073	71.75	0%	0%
Log Apply	15.23	42.15	14.66	40.31	-4%	-4%	15.02	41.25	2%	2%
Translat							0.005	0.177		

Figure 9-11 RECOVER TABLESPACE list of 100 partitions on a source versus redirected target

- ▶ Scenario 4: Recover a table space list of 100 partitions by using the **TOLOGPOINT** recovery option.
  - Run a **RECOVER TABLESPACE** list of 100 partitions with **TOLOGPOINT** on the source table space by using the Db2 12 base code level.
  - Run a **RECOVER TABLESPACE** list of 100 partitions with **TOLOGPOINT** on the source table space by using a code level that supports redirected recovery.
  - Run a **RECOVER TABLESPACE** list of 100 partitions with **TOLOGPOINT** from the source table space to the redirected target table space by using a code level that supports redirected recovery.

As shown in Figure 9-12 on page 297, no performance degradation is observed when the results of running **RECOVER TABLESPACE** with **TOLOGPOINT** on a list of 100 partitions by using the base code library are compared against the results of using a code level that supports redirected recovery.

Comparing the results of running **RECOVER TABLESPACE** with **TOLOGPOINT** on the source table space list of 100 partitions (at base code level) against the results of running a redirected recovery with **TOLOGPOINT** on the target table space list of 100 partitions (by using a code level that supports redirected recovery) shows an overall similar CPU time and elapsed time.



Table space of 100 millions rows and 100 partitions, 2 indexes 10 million rows were updated	Source						Redirected			
	Db2 12 base		With Db2 12 APARs		delta %Diff (base vs APARs)		With Db2 12 APARs		delta %Diff (base vs APARs)	
	CPU	Elapsed time	CPU	Elapsed time	CPU	Elapsed time	CPU	Elapsed time	CPU	Elapsed time
RECOVER LIST OF 100 PARTITIONS TOLOGPOINT LOGAPPLY PHASE: • 25,883,335 LOG RECORDS READ • 12,941,200 LOG RECORDS APPLIED LOGCSR PHASE: • 71,709 LOG RECORDS READ • 0 LOG RECORDS APPLIED										
<b>TOTAL Job</b>	22.47	117.82	21.62	114.89	-4%	-2%	22.52	113.95	4%	-1%
Restore Read	1.936	73.22	1.932	71.30	0%	-3%	2.195	71.36	14%	0%
Restore Write	3.036	73.22	3.023	71.30	0%	-3%	3.043	71.36	1%	0%
Log Apply	15.49	43.86	14.81	43.02	-4%	-2%	15.53	41.95	5%	-2%
Logcrrs	0.037	0.139	0.037	0.13	0%	-6%	0.039	0.143	6%	9%
Translat							0.006	0.179		

Figure 9-12 RECOVER TABLESPACE with TOLOGPOINT on a source versus a redirected target

### Scenarios and results for a redirected recovery of indexes

The following four scenarios were used to evaluate the performance improvements that are associated with the redirected recovery of indexes. Measurements are provided for both PIs and NPIs.

- ▶ Scenario 1: Recover an NPI to the current state.
  - Run **RECOVER INDEXSPACE** on the source index space by using the Db2 12 base code level.
  - Run **RECOVER INDEXSPACE** on the source index space by using a code level that supports redirected recovery.
  - Run **RECOVER INDEXSPACE** from the source index space to the redirected target index space by using a code level that supports redirected recovery.

Figure 9-13 shows that no performance degradation is observed when the results of running **RECOVER INDEXSPACE** by using the base code library are compared against the results of using a code level that supports redirected recovery.

Comparing the results of running **RECOVER INDEXSPACE** on the source NPI (by using the base code level) against the results of running a redirected **RECOVER INDEXSPACE** from the source NPI to the target NPI (by using a code level that supports redirected recovery) shows a similar overall CPU time and elapsed time.

NPI	Recover source index space						Redirect recover target index space (RR)			
	Db2 12 base		with Db2 12 APARs		%Diff (base vs APARs)		with Db2 12 APARs		%Diff (base vs APARs)	
	CPU (s)	Elapsed (s)	CPU (s)	Elapsed (s)	CPU%	Elapsed %	CPU (s)	Elapsed (s)	CPU%	Elapsed %
Table space of 100 millions rows and 100 partitions 10 million rows were updated <b>RECOVER single index space to current</b> base: LOGAPPLY PHASE: 40341206 LOG RECORDS READ, 19351777 LOG RECORDS APPLIED EP RC: LOGAPPLY PHASE: 29096071 LOG RECORDS READ, 18440577 LOG RECORDS APPLIED EP RR: LOGAPPLY PHASE: 29096071 LOG RECORDS READ, 18440577 LOG RECORDS APPLIED										
Restore Read	0.88	32.62	0.89	33.01	0.4%	1.2%	0.89	31.98	0.4%	-2.0%
Restore Write	1.36	32.62	1.37	33.01	0.5%	1.2%	1.37	31.98	1.0%	-2.0%
Log Apply	19.06	718.47	18.34	638.21	-3.8%	-11.2%	18.32	693.58	-3.9%	-3.5%
Translat							0.00	0.00		
<b>Total Job</b>	21.32	751.45	20.60	671.56	-3.4%	-10.6%	20.59	725.99	-3.4%	-3.4%

Figure 9-13 Performance of RECOVER NPI index on a source versus a redirected target

- ▶ Scenario 2: Recover an NPI by using the TORBA recovery option.
  - Run **RECOVER INDEXSPACE** on the source index space **TORBA** by using the Db2 12 base code level.
  - Run **RECOVER INDEXSPACE** on the source index space **TORBA** by using a code level that supports redirected recovery.
  - Run **RECOVER INDEXSPACE** from the source index space to the redirected target index space **TORBA** by using a code level that supports redirected recovery.

As shown in Figure 9-14, no performance degradation is observed when the results of running **RECOVER INDEXSPACE TORBA** by using the base code library are compared against the results of using a code level that supports redirected recovery.

Comparing the results of running **RECOVER INDEXSPACE TORBA** on the source NPI (base code level) against the results of running a redirected recovery of the index space with **TORBA** on the target NPI (by using a code level that supports redirected recovery) shows an overall 8.9% variation in CPU time and 8.8% variation in elapsed time. The 12.5% CPU and 18.4% elapsed time difference in the LOGUNDO phase is because redirected recovery does not write compensation log records during the LOGUNDO phase.

NPI TORBA Table space of 100 millions rows and 100 partitions 10 million rows were updated RECOVER single index space TORBA LOGAPPLY PHASE: 23106131 LOG RECORDS READ, 12455662 LOG RECORDS APPLIED LOGCSR PHASE: 1265 LOG RECORDS READ, 0 LOG RECORDS APPLIED LOGUNDO PHASE: 26194248 LOG RECORDS READ, 5984915 LOG RECORDS APPLIED	Recover source index space						Redirect recover target index space (RR)			
	Db2 12 base		with Db2 12 APARs		%Diff (base vs APARs)		with Db2 12 APARs		%Diff (base vs APARs)	
	CPU (s)	Elapsed (s)	CPU (s)	Elapsed (s)	CPU%	Elapsed%	CPU (s)	Elapsed (s)	CPU%	Elapsed%
Restore Read	0.88	32.15	0.89	31.99	0.6%	-0.5%	0.89	32.04	0.6%	-0.3%
Restore Write	1.36	32.15	1.37	31.99	0.7%	-0.5%	1.37	32.04	0.7%	-0.3%
Log Apply	14.01	407.69	14.13	416.96	0.8%	2.3%	13.99	405.76	-0.1%	-0.5%
Log CSR	0.00	0.00	0.00	0.00	0.4%	0.0%	0.00	0.00	3.5%	0.0%
LogUNDO	39.68	379.02	39.63	400.77	-0.1%	5.7%	34.71	309.15	-12.5%	-18.4%
Translat							0.00	0.00		
<b>Total Job</b>	<b>55.95</b>	<b>819.22</b>	<b>56.04</b>	<b>850.09</b>	<b>0.2%</b>	<b>3.8%</b>	<b>50.99</b>	<b>747.40</b>	<b>-8.3%</b>	<b>-8.8%</b>

Figure 9-14 Performance of RECOVER NPI index TORBA on a source versus a redirected target

- ▶ Scenario 3: Recover a PI to the current state.
  - Run **RECOVER INDEXSPACE** on the source index space by using the Db2 12 base code level.
  - Run **RECOVER INDEXSPACE** on the source index space to the current state by using a code level that supports redirected recovery.
  - Run **RECOVER INDEXSPACE** from the source index space to the redirected target index space by using a code level that supports redirected recovery.

As shown in Figure 9-15, no performance degradation is observed when the results of recovering a PI to the current state by using the base code library are compared against the results of using a code level that supports redirected recovery.

Comparing the results of recovering a PI on the source index space (base code level) against the results of a redirected recovery of a PI on the target index (by using a code level that supports redirected recovery) from the source PI shows an overall 7.1% variation in CPU time and an overall 5.1% variation in elapsed time.

PI Table space of 100 millions rows and 100 partitions 10 million rows were updated RECOVER single index space to current base: LOGAPPLY PHASE: 40341225 LOG RECORDS READ, 13588689 LOG RECORDS APPLIED EP RC: LOGAPPLY PHASE: 29098853 LOG RECORDS READ, 12707368 LOG RECORDS APPLIED	Recover source index space						Redirect recover target index space (RR)			
	Db2 12 base		with Db2 12 APARs		%Diff (base vs APARs)		with Db2 12 APARs		%Diff (base vs APARs)	
	CPU (s)	Elapsed (s)	CPU (s)	Elapsed (s)	CPU%	Elapsed%	CPU (s)	Elapsed (s)	CPU%	Elapsed%
Restore	0.21	13.11	0.21	13.74	1.4%	4.8%	0.21	12.50	2.7%	-4.6%
Log Apply	17.23	40.40	15.69	38.07	-9.0%	-5.8%	15.67	37.72	-9.1%	-6.6%
Translat							0.00	0.02		
<b>Total Job</b>	<b>17.54</b>	<b>53.72</b>	<b>16.00</b>	<b>52.07</b>	<b>-8.8%</b>	<b>-3.1%</b>	<b>16.29</b>	<b>51.00</b>	<b>-7.1%</b>	<b>-5.1%</b>

Figure 9-15 Performance of RECOVER PI index on a source versus a redirect target

- ▶ Scenario 4: Recover a PI by using the TORBA recovery option.
  - Run **RECOVER INDEXSPACE TORBA** on the source index space by using the Db2 12 base code level.
  - Run **RECOVER INDEXSPACE TORBA** on the source index space by using a code level that supports redirected recovery.
  - Run **RECOVER INDEXSPACE TORBA** from the source index space to the redirected target index space by using a code level that supports redirected recovery.

As shown in Figure 9-16, no performance degradation is observed when the results of running **RECOVER INDEXSPACE TORBA** by using the base code library are compared against the results of using a code level that supports redirected recovery.

Comparing the results of running **RECOVER TORBA** on the source PI (by using the base code level) against the results of running a redirected recovery with **TORBA** on the target PI from the source PI shows an overall 10% variation in CPU time and an overall 23% variation in elapsed time. The 15.9% CPU and 37.1% elapsed time difference in the LOGUNDO phase is because redirected recovery does not write compensation log records during the LOGUNDO phase.

PI TORBA	Recover source index space						Redirect recover target index space (RR)				
	Db2 12 base		with Db2 12 APARs		%Diff (base vs APARs)		with Db2 12 APARs		%Diff (base vs APARs)		
	CPU (s)	Elapsed (s)	CPU (s)	Elapsed (s)	CPU%	Elapsed%	CPU (s)	Elapsed (s)	CPU%	Elapsed%	
Table space of 100 millions rows and 100 partitions 10 million rows were updated <b>RECOVER single index space TORBA</b> LOGAPPLY PHASE: 23106150 LOG RECORDS READ, 6722452 LOG RECORDS APPLIED LOGCSR PHASE: 1265 LOG RECORDS READ, 0 LOG RECORDS APPLIED LOGUNDO PHASE: 26194272 LOG RECORDS READ, 5984916 LOG RECORDS APPLIED											
Restore	0.20	12.15	0.20	12.12	-0.9%	-0.2%	0.21	12.12	0.8%	-0.2%	
Log Apply	11.36	32.97	11.35	33.06	-0.1%	0.3%	11.39	33.15	0.2%	0.6%	
Log CSR	0.00	0.00	0.00	0.00	-0.8%	0.0%	0.00	0.00	2.7%	0.0%	
Log UNDO	22.65	79.07	22.70	79.19	0.2%	0.2%	19.05	49.76	-15.9%	-37.1%	
Translat							0.00	0.02			
<b>Total Job</b>	<b>34.49</b>	<b>124.63</b>	<b>34.58</b>	<b>124.90</b>	<b>0.3%</b>	<b>0.2%</b>	<b>31.03</b>	<b>95.75</b>	<b>-10.0%</b>	<b>-23.2%</b>	

Figure 9-16 Performance of RECOVER PI index TORBA on a source versus a redirected target

### Summary of results

The elapsed and CPU time when using **RECOVER** from the source to the target table space or index space are similar to the elapsed time and CPU time of recovery on the source object when the object is recovered to the current state. However, you might experience some increase in CPU time and elapsed time when recovering the source object to an earlier point-in-time when there are many log records to undo. These increases are because redirected point-in-time recovery on the target object does not write compensation log records during the LOGUNDO phase when backing out the uncommitted work, but recovery to the source object does.

A new utility phase, TRANSLAT, is added to perform conversion of the OBIDs in the pages of the target objects from the source to the target. The cost of the TRANSLAT phase is small.

### 9.5.3 Conclusion

Redirected recovery provides Db2 system programmers and database administrators a useful method to test recoveries on the production system without affecting production objects and applications.

## 9.6 Migrating a multi-table table space to PBG universal table spaces

Db2 deprecated support for simple table spaces and segmented table spaces. For existing simple and segmented table spaces, a best practice is to convert them to universal table spaces (UTSs). Db2 10 introduced a method for converting a single-table simple or segmented table space to a PBG table space as a pending definition change by using an **ALTER TABLESPACE** statement with the **MAXPARTITIONS** option. However, the process for migrating multi-table simple or segmented table spaces is much more difficult because you must unload tables; drop and re-create them; re-create all the dependent objects and referential relationships; grant authorization privileges; and load the data back into the newly defined objects. This process also requires an application outage.

This section describes a method for moving the tables from a multi-table table space to separate PBG table spaces, with minimal impact and without having to regrant authorization privileges and re-create dependent objects.

To simplify the process of migrating multi-table or segmented table spaces, the **ALTER TABLESPACE** statement is enhanced with a **MOVE TABLE** option. The **MOVE TABLE** option enables you to specify the name of the table that is to be moved and the name of the target PBG table space.

To leverage this process, the target PBG table space must be created with **DEFINE NO** and **MAXPARTITIONS 1**. When a table is moved this way, all existing authorization privileges, object dependencies, referential relationships, and associated objects are retained and unaffected. If the data set of the source table space exists (is defined), the alter is run as a pending definition change that is materialized by a subsequent online **REORG TABLESPACE** with either **SHRLEVEL REFERENCE** or **CHANGE**. Otherwise, the alter is an immediate change. Dependent packages are invalidated only for the moved tables. If not all tables are moved, packages that depend on the tables that remain in the (original) source table space are *not* invalidated.

### 9.6.1 Requirements

To use this feature, your Db2 environment must meet the following minimum requirements:

- ▶ Db2 12 at FL 508 or later or Db2 13
- ▶ For Db2 12, APPLCOMPAT level 508 or later
- ▶ For Db2 12, APAR PH29392

### 9.6.2 Performance measurement

This section describes the measurements that were taken to assess the performance effects that are associated with migrating multi-table table spaces to PBG UTS by using the **MOVE TABLE** option.

#### Measurement environment

The performance evaluation was conducted on IBM z14 hardware with four general processors, no zIIPs, and running z/OS 2.3 and Db2 12.

## Measurement scenario description

An individual **ALTER TABLESPACE** statement can specify only a single table to be moved. However, multiple **ALTER TABLESPACE** statements for the same table space can be run, and a single subsequent **REORG TABLESPACE** can materialize them all concurrently. The focus of these performance tests is to determine how expensive the **REORG TABLESPACE** is with a varying number of pending move tables, and to compare the results to a **REORG TABLESPACE** with no pending move table.

A **REORG TABLESPACE SHRLEVEL CHANGE** runs (and measured) by using the following five different sets of tables in the source segmented table space:

- ▶ Ten tables in which each table has 10 million rows and no index
- ▶ Twenty tables in which each table has 5 million rows and no index
- ▶ Fifty tables in which each table has 2 million rows and no index
- ▶ One hundred tables in which each table has 1 million rows and no index
- ▶ Five hundred empty tables with no index

## Results

The measurement results are shown in Figure 9-17, Figure 9-18, and Figure 9-19 on page 302.

REORG TABLESPACE SHRLEVEL CHANGE	Db2 12 BASE		With Db2 12 APAR		%Delta (Base vs APAR)	
	CPU	Elapsed time	CPU	Elapsed time	CPU	Elapsed time
10 tables, each table has 10 million rows	76.11	168.35	73.33	167.65	-4%	0%
20 tables, each table has 5 million rows	76.61	170.4	74.32	170.12	-3%	0%
50 tables, each table has 2 million rows	74.41	171.33	74.17	170.72	0%	0%
100 tables, each table has 1 million rows	75.2	171.5	74.14	172.16	-1%	0%
500 empty tables	0.15	1.61	0.15	1.41	0%	-12%

Figure 9-17 REORG TABLESPACE of the source table space between base and APAR

REORG TABLESPACE SHRLEVEL CHANGE	Source (no move table)		Moved all tables		Delta % Diff (all tables moved vs no move table)	
	CPU	Elapse time	CPU	Elapse time	CPU	Elapse time
10 tables, each table has 10 million rows	66.75	140.77	67.58	137.38	1%	-2%
20 tables, each table has 5 million rows	67.31	141.67	64.69	138.8	-4%	-2%
50 tables, each table has 2 million rows	66.5	142.72	70.8	144.8	6%	1%
100 tables, each table has 1 million rows	63.04	139.68	74	149.62	17%	7%
500 empty tables	0.17	1.97	2.58	68.42	1418%	3373%

Figure 9-18 REORG TABLESPACE with pending move tables versus no pending move table

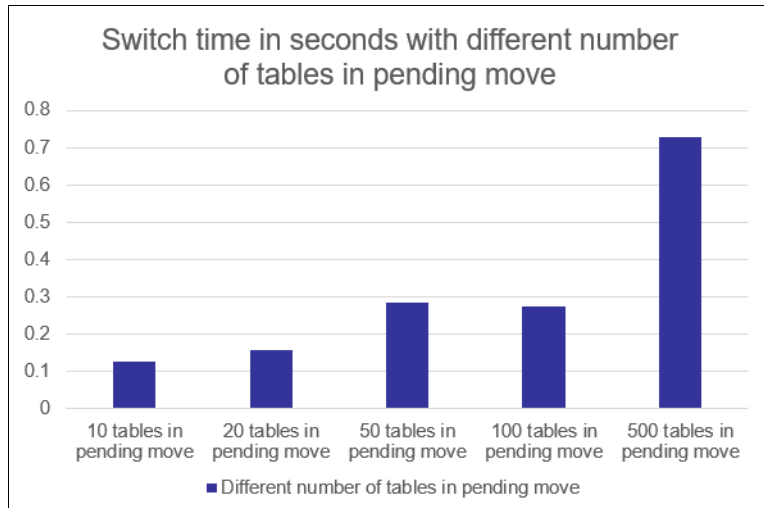


Figure 9-19 REORG switch time with different numbers of tables in pending move

### Summary of results

The following list summarizes the key points of this performance assessment:

- ▶ No performance regression is observed when the results of running **REORG TABLESPACE** by using a code level that supports the **MOVE TABLE** option are compared to the Db2 12 base code level.
- ▶ CPU and elapsed time increase when the number of tables in pending move is 50 or more. The increase in elapsed time is mainly class 3 time for open and close and delete, rename, and define of shadow data sets.
- ▶ The **REORG** switch time, which directly affects the duration of the outage during **REORG** materialization, increases when the number of tables in the “move pending” status increases.

### 9.6.3 Usage considerations

Unlike other pending definition changes, a **REORG** that materializes an **ALTER TABLESPACE MOVE TABLE** does not allow inline statistics to be gathered. A subsequent **RUNSTATS** or inline statistics run must be scheduled.

When many tables are moved in a single **REORG**, each table results in more shadow data sets being created and opened during the **REORG**. If many tables are being moved in a single **REORG**, you might encounter the **DSMAX** limitation (the **DSMAX** subsystem parameter controls the maximum number of data sets that can be open at one time).

### 9.6.4 Conclusion

UTS is the strategic table space type, and this feature provides a convenient method for moving existing tables from multiple table non-UTS table spaces to individual PBG table spaces with minimal impact.

## 9.7 REORG INDEX NOSYSUT1

Throughout the last several versions of Db2 for z/OS, the **REORG INDEX** utility infrastructure has not changed significantly. It has mainly remained a serial and single-threaded utility operation from start to end. Most of the processing is spent unloading the index keys from the target index or partition, writing out these keys to a temporary work data set (default SYSUT1), and then rereading these keys to build the target index. From a resource usage perspective, the workspace requirement and elapsed time of the **REORG INDEX** utility increases linearly as the size of the index increases, which is increasingly challenging for database administrators to manage as the size of database objects continues to grow.

This enhancement introduces a new **NOSYSUT1** keyword. It is intended to improve the performance and resource consumption of the **REORG INDEX** utility with **SHRLEVEL REFERENCE** or **CHANGE** in two ways:

- ▶ It eliminates the requirement of the SYSUT1 work data set to hold the unloaded keys before loading them back.
- ▶ It enables utility subtask parallelism to perform unloading and building of index keys per partitions in parallel.

**REORG INDEX** with **SHRLEVEL NONE** is not affected by this new behavior.

### 9.7.1 Requirements

To use this feature, your Db2 environment must meet the following minimum requirements:

- ▶ Db2 12 at FL 100 or later.
- ▶ APAR PH25217.
- ▶ Either the **NOSYSUT1** keyword must be specified on the **REORG INDEX** statement, or the **REORG\_INDEX\_NOSYSUT1** subsystem parameter must be set to YES (NO is the default setting).

If you want to use Db3 13, you must use the following function-level-related considerations:

- ▶ If you are using Db2 13 at FL 100, either the **NOSYSUT1** keyword must be specified on the **REORG INDEX** statement, or the **REORG\_INDEX\_NOSYSUT1** subsystem parameter must be set to YES (NO is the default setting).
- ▶ If you are using Db2 13 at FL 500 or later, **REORG INDEX SHRLEVEL REFERENCE** or **CHANGE** enforces the **NOSYSUT1** option by default, which becomes the only available behavior starting at FL 500.

### 9.7.2 Performance measurements

This section describes the measurements that were taken to assess the performance effects of using the **REORG INDEX NOSYSUT1** keyword.

#### Measurement environment

The performance measurements were conducted on an IBM z14 hardware with six general CPs, three zIIPs, and 56 GB of real storage, and running z/OS 2.3 and Db2 12.

## Measurement scenario description

Because this enhancement is intended to improve the performance of the unload and build phase of the **REORG INDEX** utility, the test focuses on these two phases without log apply. The TPCCH Lineltem table was used for these tests. It has 30 LPARs. The total number of records is approximately 180 million. The table is populated by the **LOAD** utility.

The indexes that are evaluated include PI, DPSI, NPI, NPI defined with multiple columns, and NPI defined with a mix of char and varchar columns. **REORG INDEX** statements are either using **SHRLEVEL REFERENCE** or **SHRLEVEL CHANGE**.

## Results

The performance results in Figure 9-20 show that with this enhancement the elapsed time improves up to 86% and the CPU + zIIP time improves up to 61%. With this enhancement, **REORG INDEX** is up to 99% zIIP eligible.

	Db2 12 base	Db2 12 base + new code	
		NOSYSUT1	
Index	Elapsed time	Elapsed time	%delta
PI	38.2	5.14	-86.54
NPI	27.62	13.37	-51.59
DPSI	34.23	4.54	-86.74
NPI (multi cols)	32.26	16.49	-48.88
NPI (char+varchar cols)	140.29	65.26	-53.48
Index	CPU Time	CPU Time	%delta
PI	11.45	3.94	-65.59
NPI	10.95	0.01	-99.91
DPSI	10.95	3.44	-68.58
NPI (multi cols)	11.68	0.01	-99.91
NPI (char+varchar cols)	26.48	0.01	-99.96
Index	CPU +zIIP time	CPU +zIIP time	%delta
PI	17.08	9.38	-45.08
NPI	16.05	7.05	-56.07
DPSI	16.02	8.08	-49.56
NPI (multi cols)	17.21	7.66	-55.49
NPI (char+varchar cols)	39.02	16.69	-57.23

Figure 9-20 Performance comparison for REORG INDEX with and without NOSYSUT1

This feature introduced approximately 40 MB of real storage usage overhead, which is observed in the evaluation test cases.



## Summary of results

This new feature improves the elapsed time of **REORG INDEX SHRLEVEL REFERENCE** or **CHANGE** along with a reduction in CPU time, especially when zIIP capacity is available, at the expense of the minimal increase of real storage usage. With this performance improvement, the **NOSYSUT1** behavior becomes the only behavior after Db2 13 FL V13R1M500.

### 9.7.3 Monitoring information

The following new message is added to support the new feature:

```
DSNU2909I csect-name – INDEX OR INDEX PARTITIONS WILL BE UNLOADED AND BUILT IN  
PARALLEL, NUMBER OF TASKS=n
```


### 9.7.4 Usage considerations

In Db2 12 and Db2 13 at V13R1M100, the default setting of the **REORG\_INDEX\_NOSYSUT1** subsystem parameter is **NO**, which means that to use this feature that you must either explicitly specify the **NOSYSUT1** keyword on the **REORG INDEX** statements, or set the **REORG\_INDEX\_NOSYSUT1** subsystem parameter to **YES**. Starting at Db2 13 FL 500, Db2 automatically uses this feature by default and it cannot be changed. This enhancement does increase some real storage usage.

### 9.7.5 Conclusion

This new feature improves the elapsed time (up to 86%) and CPU time (up to 61%) of **REORG INDEX SHRLEVEL REFERENCE** or **CHANGE**. It also improves the zIIP eligibility (up to 99%) for **REORG INDEX**.





## Performance enhancements for IDAA for z/OS and IBM Db2 for z/OS Data Gate

Since the release of Db2 12, IBM Db2 Analytics Accelerator (IDAA) for z/OS has undergone many changes and improvements. One of the biggest changes was the migration from IBM Netezza® machines in IDAA 5 to either IBM Integrated Analytics Systems (IAS) hardware or configured logical partitions (LPARs) on the IBM Z platform in Version 7. The new Version 7 platforms use Db2 Warehouse as a back-end engine to accelerate queries instead of the Field Programmable Gate Array (FPGA)-based architecture, which was used in Version 5.

As expected, the significant migration in hardware from Version 5 to Version 7 is accompanied by an abundance of features and improvements in IDAA for z/OS V7. This chapter documents the following enhancements:

- ▶ IBM Integrated Synchronization
- ▶ Cluster load
- ▶ COLJOIN 1 enablement
- ▶ Direct I/O for IBM Db2 Analytics Accelerator on IBM Z
- ▶ Automatic statistics collection and collection improvement
- ▶ Performance impact of AT-TLS enablement

Additionally, IBM Db2 for z/OS Data Gate was recently introduced, which provides Db2 for z/OS users with a modern approach to accelerating queries in hybrid cloud environment. As with IDAA for z/OS, with IBM Db2 for z/OS Data Gate, you can redirect queries away from IBM zSystems, but instead of running within an IDAA for z/OS system, IBM Db2 for z/OS Data Gate allows those queries to run in a hybrid cloud environment. Performance results and other information about IBM Db2 for z/OS Data Gate are provided in 10.8, “Performance overview for IBM Db2 for z/OS Data Gate” on page 323.

Unless otherwise stated, the results that are provided were generated by using a Db2 12 environment. Comparisons with Db2 13 are provided for specific workloads in 10.6, “Query workload comparison between Db2 12 and Db2 13 for z/OS” on page 321

For more information about the query workloads that were used in this chapter, see Appendix C, “IBM Db2 Analytics Accelerator for z/OS workloads” on page 421.

## 10.1 IBM Integrated Synchronization

In IDAA for z/OS V7.5, a new replication protocol between Db2 for z/OS and the accelerator that is called IBM Integrated Synchronization was introduced in addition to the existing data replication support that is provided by IBM InfoSphere® Change Data Capture (CDC). IBM Integrated Synchronization is a built-in IDAA for z/OS feature that you use to reduce the CPU costs that are associated with replicating Db2 for z/OS transactional data on the mainframe. IBM Integrated Synchronization provides the following advantages:

- ▶ It is bundled with IDAA for z/OS, which means that no extra software is needed.
- ▶ It provides simplified administration, packaging, upgrades, and support.
- ▶ It achieves better replication performance in end-to-end replication latency and average replication throughput with a streamlined, optimized design in which most of the processing is done on the accelerator.
- ▶ It reduces z/OS CPU costs by taking full advantage of IBM zSystems Integrated Information Processors (zIIP).

### 10.1.1 Requirements

To use the IBM Integrated Synchronization feature, your Db2 environment must meet the following minimum requirements:

- ▶ Db2 12 at function level (FL) V12R1M500 or later or Db2 13
- ▶ APAR PH06628 (PTF UI63356) applied
- ▶ IDAA for z/OS 7.5 or later

### 10.1.2 Performance measurement

This section describes the measurements that were taken to assess the performance of IBM Integrated Synchronization compared to the existing InfoSphere CDC replication functions.

#### Measurement environment

The environment in which the performance evaluation was conducted consists of the following components:

- ▶ IBM z14 hardware with six general-purpose central processors (GPCs) and two zIIPs
- ▶ 96 GB of memory
- ▶ 10 GB network to the accelerator
- ▶ z/OS 2.2
- ▶ Db2 12 for z/OS
- ▶ IDAA 7.5

- ▶ M4002-010 (IAS full-rack) Accelerator hardware

### Measurement scenario

This section describes the usage scenarios that were used to evaluate the performance of IBM Integrated Synchronization:

- ▶ The test workload uses 500 partitioned tables with concurrent insert and update activity from 250 threads. The number of rows that are inserted and updated per commit is 20 rows. The maximum Db2 transaction rates, which are driven by a distributed Java program, are as follows:
  - 76.4 K rows per second inserted.
  - 76.4 K rows per second updated.
- ▶ The total amount of log read activity (based on the Db2 statistics trace) for inserts and updates:
  - **READS SATISFIED-OUTPUT BUFF:** 311.0 million.
  - **READS SATISFIED-ACTIVE LOG:** 2626.3 K.
  - **LOG RECORDS READ:** 285 086 600.00.
- ▶ The following test scenarios were run, measured, and analyzed:
  - Capture and apply with concurrent Db2 transactions.
  - Capture and apply the pre-populated log.

### Results

This section describes the results of both scenarios.

#### ***Performance summary for capture and apply with concurrent Db2 transactions***

Comparing IBM Integrated Synchronization to traditional CDC reveals the following general results:

- ▶ IBM Integrated Synchronization provides a reduction in maximum replication latency (see Figure 10-1).

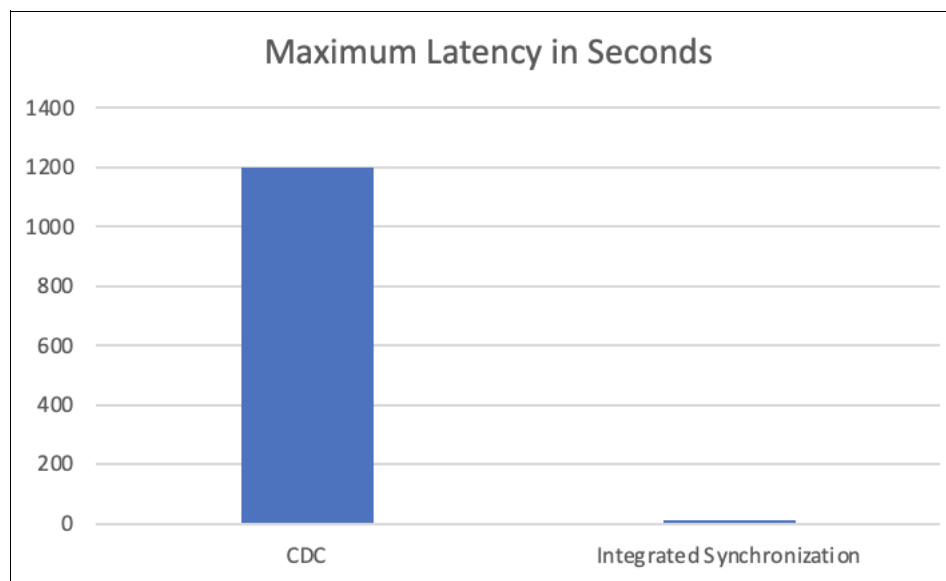


Figure 10-1 IBM Integrated Synchronization maximum latency compared to CDC

- ▶ IBM Integrated Synchronization log reading is counted in the Db2 DIST address space (see Figure 10-2).

	CDC				Integrated Synchronization			
	CP	IIP	Projected zIIP Eligible CPU	zIIP Eligible %	CP	IIP	Projected zIIP Eligible CPU	zIIP Eligible %
CDC	2401.95	0.00	0.00	0.00%	0.00	0.00	0.00	0.00%
DBM1	70.90	44.13	45.93	39.93%	74.71	45.44	49.00	40.78%
DFFWORK	2321.77	1422.39	2271.17	60.66%	2651.80	1265.82	2395.65	61.15%
DIST	158.21	0.74	0.74	0.47%	509.98	629.85	972.12	99.03%
MSTR	92.21	24.41	32.49	27.86%	59.09	14.35	18.95	25.81%
PAAGENT	0.05	0.00	0.00	0.00%	0.08	0.00	0.00	0.00%
TCPIP	58.39	0.00	0.00	0.00%	44.08	0.00	0.00	0.00%
<b>TOTAL</b>	<b>5103.52</b>	<b>1491.66</b>	<b>2350.33</b>	<b>35.64%</b>	<b>3339.74</b>	<b>1955.46</b>	<b>3435.72</b>	<b>64.88%</b>

Figure 10-2 CPU time in seconds for capture and apply with concurrent Db2 transactions

- ▶ The maximum zIIP eligibility in the DIST address space is 99%. (see Figure 10-2).
- ▶ A concurrent influx of new changes (from concurrently running transactions) makes it difficult to isolate the replication CPU usage for load-reading tasks.

**Performance summary for capture and apply with pre-populated Db2 transactions**

Capturing and applying the log data from a “pre-populated” set of transactions reveals the following general results, which are illustrated in Figure 10-3:

- ▶ IBM Integrated Synchronization uses approximately 46% of the amount of CPU time compared to CDC for the same work (CDC used 2114 seconds, but IBM Integrated Synchronization used 976 seconds).
- ▶ The combined IBM Integrated Synchronization work across all address-spaces is 96% zIIP eligible.

	CDC				Integrated Synchronization			
	CP	IIP	Projected zIIP Eligible CPU	zIIP Eligible %	CP	IIP	Projected zIIP Eligible CPU	zIIP Eligible %
CDC	2094.81	0.00	0.00	0.00%	0.00	0.00	0.00	0.00%
DBM1	0.23	0.23	0.23	48.80%	4.19	0.37	0.37	8.16%
DFFWORK	5.45	0.10	0.10	1.92%	0.00	61.36	61.36	100.00%
DIST	0.11	0.00	0.74	0.00%	1.43	874.82	874.88	99.84%
MSTR	7.78	2.35	2.35	23.24%	5.04	3.48	3.48	40.83%
PAAGENT	0.06	0.00	0.00	0.00%	0.03	0.00	0.00	0.00%
TCPIP	3.36	0.00	0.00	0.00%	26.02	0.00	0.00	0.00%
<b>TOTAL</b>	<b>2111.81</b>	<b>2.68</b>	<b>2.68</b>	<b>0.13%</b>	<b>36.71</b>	<b>940.02</b>	<b>940.09</b>	<b>96.25%</b>

Figure 10-3 CPU time in seconds for capture and apply with pre-populated Db2 logs

## Summary of results

The results of the testing revealed the following key highlights:

- ▶ Total CPU (central processor (CP) and IIP) time was reduced by up to 46% for log reading (in the pre-populated logs scenario) compared to CDC.
- ▶ IBM Integrated Synchronization log reading is 99% zIIP eligible (included in the Db2 DIST address space).
- ▶ Replication latency was reduced and replication throughput was improved.

### 10.1.3 Monitoring information

The IBM Integrated Synchronization log reader CPU time is captured in the Db2 statistics report in both the Accelerator section for a single Db2 subsystem and in the general CPU time section, as shown in Figure 10-4.

CPU TIME FOR LOG READER TASK	20:04.627643
ZIIP TIME FOR LOG READER TASK	0.000000
ZIIP ELIGIBLE TIME FOR LRT	20:04.627643

Figure 10-4 IBM Integrated Synchronization log reader CPU time in the Db2 statistics report

## 10.2 Cluster load

IDAA for z/OS 7.1.9 introduced a load improvement method in the IDAA for z/OS server to increase load throughput when loading data from Db2 for z/OS subsystems to multiple-node accelerators.

A multiple-node accelerator has a node cluster that consists of a head node and several worker nodes (also known as data nodes). Each node is running multiple database partitions, which are multiple logical nodes (MLNs). Data that is received from the Db2 for z/OS subsystem is loaded into the accelerator back-end Db2 Warehouse database by using **INSERT** statements. Before this enhancement was available, the accelerator server that is connected to the head node's MLNs was used for the load **INSERT** statements. The CPU utilization on the head node is much higher than the worker nodes (the worker nodes use little CPU). To avoid excessive CPU usage from load activities on the head node, the accelerator server has an internal logic to halt load activities when the head node's CPU utilization reaches 80%, which limits the load throughput. This experience was observed in many internal and customers' large table loads in the past.

With the cluster load feature, the accelerator server connects to every MLN on all the nodes for the load **INSERT** statements, which means that the workload on the head node is reduced and more balanced among the nodes in the cluster. With less work being directed to the head node, the CPU utilization on the head node is reduced, which results in fewer occurrences of load activities being halted and an improvement in load throughput.

Because the cluster load feature reduces the CPU usage on the head node, the accelerator server can possibly take more parallel connections between the Db2 for z/OS subsystem, and the accelerator to leverage the released head node CPU capacities to further improve load throughput. In the following tests, the stored procedure Workload Manager (WLM) variable **AQT\_MAXUNLOAD\_IN\_PARALLEL** is increased from 10 to 20, and the load throughput is improved as a result of using IDAA for z/OS 7.1.9 with cluster load compared to IDAA for z/OS 7.1.8.

## 10.2.1 Requirements

This enhancement is available in IDAA for z/OS 7.1.9 or later. No specific Db2 version or FL is required.

## 10.2.2 Performance measurement

This section describes the measurements that were taken to assess the performance of the cluster load feature.

### Measurement environment

The performance evaluation was conducted on IBM z14 hardware with six general processors, one zIIP, 80 GB of memory, and running z/OS 2.3 and Db2 12.

The IDAA for z/OS system is a 7-node full rack Sailfish system.

### Measurement scenario description

The following two scenarios were run to evaluate the performance of the cluster load feature of IDAA for z/OS V7.1.9:

- ▶ The first scenario involves an initial sequential load of two tables.

The two tables have different data volumes: one is 722 GB, and the other one is 3880 GB. The **AQT\_MAX\_UNLOAD\_IN\_PARALLEL** subsystem parameter is set to 20 when using cluster load, which means that 20 parallel unload tasks are performed on the Db2 for z/OS side, and 20 connections are made to the accelerator server to transfer the data. These results are compared to a non-cluster load in which **AQT\_MAX\_UNLOAD\_IN\_PARALLEL** is set to 10, and IDAA for z/OS 7.1.8 is used.

- ▶ The second scenario involves a reload of five tables (T1 - T5) that use five concurrent threads. Each thread reloads a single table.

The five tables have different data volumes. Both the cluster load and the non-cluster load are performed with IDAA for z/OS V7.1.9 with **AQT\_MAX\_UNLOAD\_IN\_PARALLEL** set to 20.

## Results

The results of both scenarios are documented in the following sections.

### **Sequential load with two different data volumes with a cluster load**

The data in Figure 10-5 compares an IDAA for z/OS 7.1.8 non-cluster load that uses **AQT\_MAX\_UNLOAD\_IN\_PARALLEL=10** to an IDAA for z/OS 7.1.9 cluster load that uses **AQT\_MAX\_UNLOAD\_IN\_PARALLEL=20**. Using a cluster load shows a 48% and 37% throughput improvement when loading the respective tables.

	V7 7.1.8 LOAD, AQT_MAX_UNLOAD_IN_PARALLEL=10		V7 7.1.9 Cluster Load, AQT_MAX_UNLOAD_IN_PARALLEL=20		
<b>LOAD DATA VOLUME=722GB</b>	CL.1 ET (s)	TB/hour	CL.1 ET (s)	TB/hour	Diff vs 7.1.8
	723	3.5	489	5.2	48%
<b>LOAD DATA VOLUME=3880GB</b>	CL.1 ET (s)	TB/hour	CL.1 ET (s)	TB/hour	Diff vs 7.1.8
	3421	3.7	2497	5.1	37%

Figure 10-5 Comparing the sequential load between a cluster load and a non-cluster load



**Five concurrent threads reloading five tables with a cluster load (each thread reloads one table)**

As shown in Figure 10-6, comparing the non-cluster load and the cluster load when both loads are using the same `AQT_MAX_UNLOAD_IN_PARALLEL` setting of 20 shows a 52% throughput improvement for the cluster load.

table name	data volume GB	No Cluster Load, AQT_MAX_UNLOAD_IN_PARALLEL=20		With Cluster Load, AQT_MAX_UNLOAD_IN_PARALLEL=20		
		total load time (s)	throughput TB/h	total load time (s)	throughput TB/h	diff
T1	115.66	681.07	0.6	397.87	1.04	73%
T2	130.34	797.38	0.58	421.79	1.11	91%
T3	160.98	755.64	0.76	443.95	1.29	70%
T4	209.94	710.13	1.05	575.66	1.3	24%
T5	578.29	1105.02	1.85	779.31	2.63	42%
Average			0.97		1.47	52%

Figure 10-6 Comparing the concurrent reload performance between a cluster load and a non-cluster load

Figure 10-7 shows that without a cluster load, the host CPU utilization rate reaches the 80% cap while the CPU utilization rate of worker nodes is low. With a cluster load, host and work nodes CPU utilization is better balanced, and host node CPU utilization no longer reaches 80%, which means that load activity no longer halts and a higher load throughput is achieved.

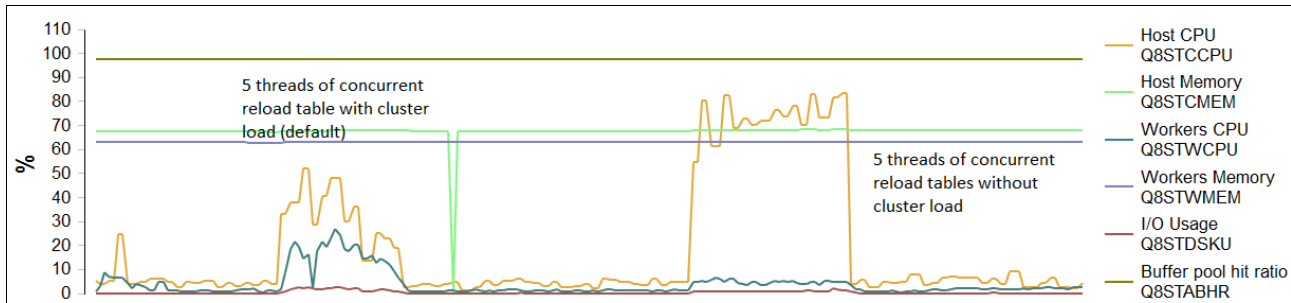


Figure 10-7 Accelerator host and worker CPU utilization comparison with versus without a cluster load

### 10.2.3 Usage considerations

This feature is enabled by default in IDAA for z/OS 7.1.9 and later. It can be deactivated through IBM Support.

### 10.2.4 Conclusion

This feature enables you to achieve a better balance among accelerator catalog nodes and other work nodes or data nodes for load or reload workloads. As a result, catalog node CPU utilization is not as much of a bottleneck as it was before. When a cluster load is used, the catalog node has more capacity to take extra parallel connections from the Db2 for z/OS side to speed up load or reload processes and improve the throughput.

## 10.3 COLJOIN 1 enablement

With the continuing pursuit of improving query performance, certain Db2 Warehouse settings are enabled by default for IDAA for z/OS usage too. One such setting is COLJOIN 1, which improved query performance in IBM internal tests.

COLJOIN 1 decompresses and expands the result set during a hash join. By doing so, the memory usage increases because the uncompressed data must be processed, but there is faster processing of the join overall. Testing revealed that the increase of memory usage did not impact the performance results.

### 10.3.1 Requirements

This enhancement is available in IDAA for z/OS 7.5.7 or later. No specific Db2 version or FL is required.

### 10.3.2 Performance measurement

The most recent IDAA for z/OS 7.5.8 testing showed an improvement or equivalent performance when compared to previous baseline measurements.

#### Performance environment and workload

Using modified TPC-H and Customer FD workloads, the following tests were conducted on an IAS system:

- ▶ IDAA for z/OS 7.5.7 was used to collect results with COLJOIN 1 both enabled and disabled.
- ▶ The following TPC-H workloads were used:
  - Single- and multi-threaded query workloads that are submitted by DSNTEP2 with a data source size of 1 TB
  - Single- and multi-threaded query workloads that are submitted by DSNTEP2 with a data source size of 5 TB
  - Single-threaded query workload that is submitted through Distributed Java with a data source size of 5 TB
  - Multi-threaded query workload that is submitted through Distributed Java with a data source size of 1 TB
- ▶ Customer FD workloads were run by using the standard setup.

#### Results

The results of the TPC-H workload measurements are shown in Figure 10-8 on page 315, and the results of the Customer FD workload are shown in Figure 10-9 on page 315.

TPC-H Workloads	Baseline (COLJOIN 1 OFF) CL2 Elapsed Time (s)	COLJOIN 1 ON CL2 Elapsed Time (s)	% Improvement
TPC-H Single Thread Workload (1TB)	975	1010.5	-3.641%
TPC-H Single Thread Workload (5TB)	3506	3259	7.045%
TPC-H Multithread Workload (1TB)	14291	13982	2.162%
TPC-H Multithread Workload (5TB)	64856	63985	1.343%
TPC-H Single Thread Java Workload (5TB)	13578	14035	-3.37%
TPC-H Multithread Java Workload (1TB)	2894	2578	10.9%

Figure 10-8 TPC-H workload results comparing COLJOIN 1 workloads to the baseline measurements

Customer FD Workload	Baseline (COLJOIN 1 OFF) CL2 Elapsed Time (s)	COLJOIN 1 ON CL2 Elapsed Time (s)	% Improvement
Customer FD Workload	3127	3184	-1.823%

Figure 10-9 Customer FD workload results comparing COLJOIN 1 to the baseline measurements

### Results summary

The TPC-H and Customer FD results show that there are some general improvements or equivalent performance compared to the previous default settings. For the TPC-H Workload, most queries did not see regression nor improvement. However, some long-running queries had better access paths because COLJOIN 1 was enabled, which resulted in an improvement in overall elapsed time.

One of the longest running queries in the TPC-H workloads saw an improvement, which led to an improvement in elapsed time for the 5 TB data size workload. This same query was excluded from multi-threaded workloads before this setting in previous versions because it has such a long run time; therefore, the multi-threaded results do not show the same amount of improvement compared to the single-threaded results.

### 10.3.3 Usage considerations

Although these results show a general improvement in query performance, some risk is involved with this setting enabled. COLJOIN 1 uses more memory and, as such, can increase the risk of receiving of out of memory conditions, poor access plans, and other performance-related issues. If any performance impact is observed after upgrading to Version 7.5.7 or later, IBM Support can help you assess whether this setting caused this performance degradation and whether disabling the setting will resolve performance issues.

## 10.4 Direct I/O for IBM Db2 Analytics Accelerator on IBM Z

Previously, IDAA for z/OS on IBM Z used buffered I/O to retrieve information from disk when processing queries. IDAA for z/OS 7.5.6 introduced direct I/O as the default I/O for data processing. Direct I/O, as the name implies, allows for direct, unbuffered access to the disk, which provides decreased CPU overhead because data copying to the buffer cache is no longer needed.

## 10.4.1 Requirements

IDAA for z/OS 7.5.6 and later is required to use this setting.

## 10.4.2 Performance observations

Before the introduction of the direct I/O feature, we observed that a few queries consumed a large amount of TEMPSPACE while they ran. These queries often regressed heavily or returned out-of-memory errors under certain system conditions. With the introduction of direct I/O, test results show that these high TEMPSPACE-consuming queries performed consistently without errors or performance issues because of the increased amount of TEMPSPACE that was provided with direct I/O enabled.

Additionally, no adverse effects were observed after changing the I/O scheme. All queries that ran during testing performed on-par or better with direct I/O enabled.

## 10.5 Automatic statistics collection and collection improvement

After a table is initially loaded or reloaded on an accelerator, statistics must be collected on the accelerator back end so that the optimizer can choose a good access path. There are several ways to collect these statistics:

- ▶ *Early stats*, which are automatic statistics that are run immediately after an initial load. This process has two rounds: the first round uses a lower sampling rate, and the second round uses a higher sampling rate.
- ▶ *Copy stats*, which copy the previous statistics into the new table version during a full table reload.
- ▶ *Auto stats*, which are collected by the Db2 Warehouse statistics daemon that runs approximately every 2 hours if a table has enough changes.
- ▶ *Real-time stats*, which are collected by the Db2 Warehouse engine to create “fabricated” stats during query runs if it determines that important statistics are missing.

### 10.5.1 Early statistics collection

When you are trying to determine the root cause of poor-performing queries, the solution often is to collect better table statistics. These statistics help the Db2 Warehouse Optimizer to identify better access paths for faster and more efficient query runs.

Because of the necessity of statistics for better query runs, the collection of statistics becomes a priority. Previously, statistics were collected by using a single-thread daemon in the Db2 Warehouse engine. However, because this daemon is single-threaded, this process can take nearly 2 hours for statistics to start being collected, which does not include the additional amount of time it takes to collect the statistics. To reduce the amount of time for statistics collection, a new method was developed in which table statistics are automatically collected immediately after an initial load completes.

## Process and status monitoring

The process begins with the accelerator starting a thread in which an explicit **RUNSTATS** command is issued after the table load completes, which ensures that the statistics are collected as soon as possible, for example:

```
RUNSTATS ON TABLE <fully-qualified-backend-tablename> ON ALL COLUMNS WITH  
DISTRIBUTION ON ALL COLUMNS AND SAMPLED DETAILED INDEXES ALL
```

This process has little to no impact on concurrent tasks.

Starting in IDAA for z/OS V7.5.2, the collection process is changed to provide better plan stability over time. By using this new method, the **RUNSTATS** statement still runs with a sampling rate that is predetermined by the system. When the **RUNSTATS** statement with the computed sampling rate completes, the system checks whether the (computed) sampling rate that was used is below a minimum threshold. By default, this threshold is 30%, but it can be changed by using the **RUNSTATS\_MINIMUM\_SAMPLING\_RATE\_PERCENTAGE** subsystem parameter by an IBM Client Success agent. If the calculated sampling rate is below this threshold, a second **RUNSTATS** statement runs with the minimum sampling rate set.

The status of this statistics collection can be monitored by using the IDAA for z/OS Data Studio plug-in, in which a “warning sign” indicates that statistics are still being collected on that table and that disappears when the process finishes.

## Requirements

To leverage the original statistics collection process (without the sampling rate calculation), you must be using IDAA for z/OS 7.5.1 or later.

To use the updated early stats collection process, you must be using IDAA for z/OS 7.5.2 or later.

## Usage considerations

Although the statistics collection begins as soon as possible, it can take a few hours before the statistics collection is finished. Again, this process occurs after an initial table load, and full reloads of tables can benefit from the Copy Table Statistics feature that is described in 10.5.2, “Copy Table Statistics” on page 317.

## 10.5.2 Copy Table Statistics

Often, a full table reload is necessary in production environments for many reasons. Regardless of the reason, queries that are accelerated shortly after the full table reload tend to perform poorly due to the lack of table statistics. Additionally, the potential amount of time it can take for the Db2 Warehouse engine to collect such statistics can be several hours.

The Copy Table Statistics feature helps to alleviate this problem by saving the table’s statistics from before the reload and copying them over after the reload completes. Therefore, queries that are submitted shortly after the reload completes perform as well as before the full table reload occurred.

## Requirements

To use this feature, you must be using IDAA for z/OS 7.5.0 or later.

## Performance measurement

To validate the performance of the Copy Table Statistics feature, several baseline measurements were taken first. The term “Regular Statistics” refers to the normal statistics that are collected after a table is initially loaded, and the term “Copy Statistics” refers to the statistics that are provided from the Copy Table Statistics feature after a reload. The test scenario consists of the following steps:

1. With the Copy Table Statistics setting disabled, perform an initial load of the TPC-H LINEITEM table. After the table is loaded, wait for statistics to be collected.
2. Take a baseline measurement of the query workload’s elapsed time in which all queries are using the LINEITEM table (Perf 1 – Regular Statistics).
3. The LINEITEM table is then reloaded, and the query workload runs without waiting for statistics to be collected and without Copy Table Statistics (Perf 2 – Reloaded performance, No Copy or Regular Statistics).
4. Remove the LINEITEM table, then do another initial load of the LINEITEM table with Copy Table Statistics enabled and run the query workload again (Perf 3 – Second Regular Statistics).
5. The LINEITEM table is reloaded with Copy Table Statistics enabled, and the query workload is run and measured again (Perf 4 – Copy Table Statistics measurement).
6. Another measurement is conducted by using the query workload with normal statistics and Copy Table Statistics enabled (Perf 5 – Regular Statistics after Copy Table Statistics).

During each run, the statement cache is cleared to ensure that all runs are as clean as possible.

## Results

The results of these measurements are summarized in Figure 10-10.

V7.5.1RC2									
CL1 elapsed Local MJ2A 1TB	perf5 (after stats) D200410E	perf2 (no stats) D200410A	perf2 AP changed vs. perf5?	perf2 vs. perf5	perf4 (copied stats) D200410D	perf4 AP changed vs. perf5?	perf4 vs. perf5	perf4 vs. perf2	
T01Q01	12.52	14.75	Y	18%	12.62	N	1%	-14%	
T01Q03	58.18	58.45	N	0%	69.38	Y	19%	19%	
T01Q04	5.2	5.16	Y	-1%	5.5	N	6%	7%	
T01Q05	7.35	12.74	Y	73%	12	Y	63%	-6%	
T01Q06	1.3	1.16	N	-11%	1.36	N	5%	17%	
T01Q07	13.09	25.05	Y	91%	37.24	Y	184%	49%	
T01Q08	4.93	4.95	N	0%	4.86	N	-1%	-2%	
T01Q09	23.54	21.07	N	-10%	23.38	N	-1%	11%	
T01Q10	505.75	515.11	Y	2%	388.65	Y	-23%	-25%	
T01Q12	2.1	2.1	N	0%	2.12	N	1%	1%	
T01Q14	2.34	1.86	Y	-21%	2.34	N	0%	26%	
T01Q15	2.81	2.69	Y	-4%	2.74	N	-2%	2%	
T01Q15A	62.6	80.2	Y	28%	62.33	N	0%	-22%	
T01Q17	5.21	5.7	Y	9%	5.05	Y	-3%	-11%	
T01Q18	19.16	18.27	Y	-5%	17.59	Y	-8%	-4%	
T01Q19	2.21	2.84	Y	29%	2.43	Y	10%	-14%	
T01Q20	4.35	8.02	Y	84%	4.26	N	-2%	-47%	
T01Q21	54.96	36.42	Y	-34%	24.93	Y	-55%	-32%	
T01QBLL	1	1.08	N	8%	1.05	N	5%	-3%	
SUM	788.6	817.62		4%	679.83		-14%	-17%	
SUM-Q10	282.85	302.51		7%	291.18		3%	-4%	

Figure 10-10 Detailed query performance breakdown of all steps of the performance measurement process

Some variation exists between each run. The access path of certain queries also changed between the different settings, which resulted in these variations. Even though some regressions were observed for certain queries, the findings show that the usage of the Copy Table Statistics feature improves the query run time immediately after a reload.

### Usage considerations

The Copy Table Statistics feature works only when a full table reload occurs. Partial table reloads do not receive the benefit of the Copy Table Statistics feature.

## 10.5.3 Explicit RUNSTATS

Despite all the different statistics collection methods that are available, there are scenarios in which statistics are either not collected soon enough or are not collected at all. For example, accelerator-only tables (AOTs) are populated by **INSERT** statements, not by load or reload operations, so the Early statistics and Copy table statistics processes do not apply to AOTs, and replication-enabled tables with massive transaction sync-ups often have outdated statistics. Auto stats has a 2-hour cycle and might not collect the statistics in a timely manner.

To alleviate this situation, IDAA for z/OS 7.5.4 introduced a stored procedure that is called **ACCEL\_COLLECT\_TABLE\_STATISTICS**. By using it, you can explicitly collect statistics. You can specify multiple tables in a table set and **RUNSTATS** runs concurrently for each table, which can potentially cause increased CPU and memory usage on the accelerator.

A performance study was conducted to assess the impact of this feature on the accelerator resources and to evaluate the impact on query performance when multiple explicit **RUNSTATS** run concurrently.

### Requirements

To use this feature, you must be running IDAA for z/OS 7.5.4 or later. No specific version of Db2 for z/OS is required.

### Measurement environment

The performance evaluation was conducted on IBM z14 hardware with six general processors, one zIIP, 80 GB of memory, and running z/OS 2.3 and Db2 12.

The accelerator hardware is a 7-node full rack Sailfish system.

### Measurement scenario description

The following scenarios were run to evaluate this feature:

- ▶ The first scenario performs a sequence of statistics collection invocations with an increasing number of tables in the table set: 1 table, 5 tables, 10 tables, 20 tables, 30 tables, 40 tables, and 50 tables. The elapsed time for each table's statistics run was measured.
- ▶ The second scenario runs a statistics collection concurrently for 10 tables, and at the same time measures a query workload.

The 50 tables that are used in these tests are all AOTs with five different table definitions, data volumes, and number of records:

- ▶ Tables 1 - 10 have a data volume of 23.43 GB and contain 15,000,000 records.
- ▶ Tables 11 - 20 have data volume of 26.07 GB and contain 200,000,000 records.
- ▶ Tables 21 - 30 have a data volume of 115.66 GB and contain 800,000,000 records.

- ▶ Tables 31 - 40 have a data volume of 160.98 GB and contain 1,500,000,000 records.
- ▶ Tables 41 - 50 have a data volume of 722.15 GB and contain 5,999,989,709 records.

The following scenarios were run to evaluate this feature:

- ▶ The first scenario performs a sequence of statistics collection invocations with an increasing number of tables in the table set: 1 table, 5 tables, 10 tables, 20 tables, 30 tables, 40 tables, and 50 tables. The average elapsed time of each groups' individual table's **RUNSTATS** is used in the results.
  - One table in the table set: T41
  - Five tables in the table set: T1, T11, T21, T31, and T41
  - Ten tables in the table set: T1-T2, T21-T22, T31-T32, and T41 - T42
  - Twenty tables in the table set: T1-T4, T21-T24, T31-T34, and T41 - T44
  - Thirty tables in the table set: T1-T6, T21-T26, T31-T36, and T41 - T46
  - Forty tables in the table set: T1-T8, T21-T28, T31-T38, and T41 - T48
  - Fifty tables in the table set: T1-T10, T21-T30, T31-T40, and T41 - T50
- ▶ The second scenario runs a statistics collection concurrently for 10 tables while measuring a query workload.

### Measurement results

For the first scenario, all the explicit **RUNSTATS** were successful.

As shown in Figure 10-11, when the number of tables in the table set increases, the elapsed time of explicit **RUNSTATS** becomes longer but is still within a reasonable range. For example, the elapsed time of the explicit **RUNSTATS** for table 1 is 20.4 seconds with table set=5 and 36.6 seconds with table set=50.

#Rows	Table	elapsed time in seconds of each table's explicit runstats with different number of tables in the tableset						
		1 table	5 tables	10 tables	20 tables	30 tables	40 tables	50 tables
1.5E+08	Table 1 ~ 10		20.4	25.2	30.6	25.8	34.6	36.6
2E+08	Table 11 ~ 20		34.3	38.6	48.9	47.8	53.2	58.5
8E+08	Table 21 ~ 30		57.8	64.5	75.8	84.3	86.1	100.5
1.5E+09	Table 31 ~ 40		177.6	186.1	213.8	224.2	227.9	259.2
6E+09	Table 41 ~ 50	1107.1	1106.1	1109.4	1161.4	1202.0	1192.8	1239.2

Figure 10-11 Elapsed time in seconds of each table's explicit **RUNSTATS** with different numbers of tables in the table set

As shown in Figure 10-12 on page 321, the host CPU usage on the accelerator has spikes during the first test scenario. When table set=50 is used, the host CPU usage reaches 70%. A best practice is not to have too many explicit **RUNSTATS** running concurrently.



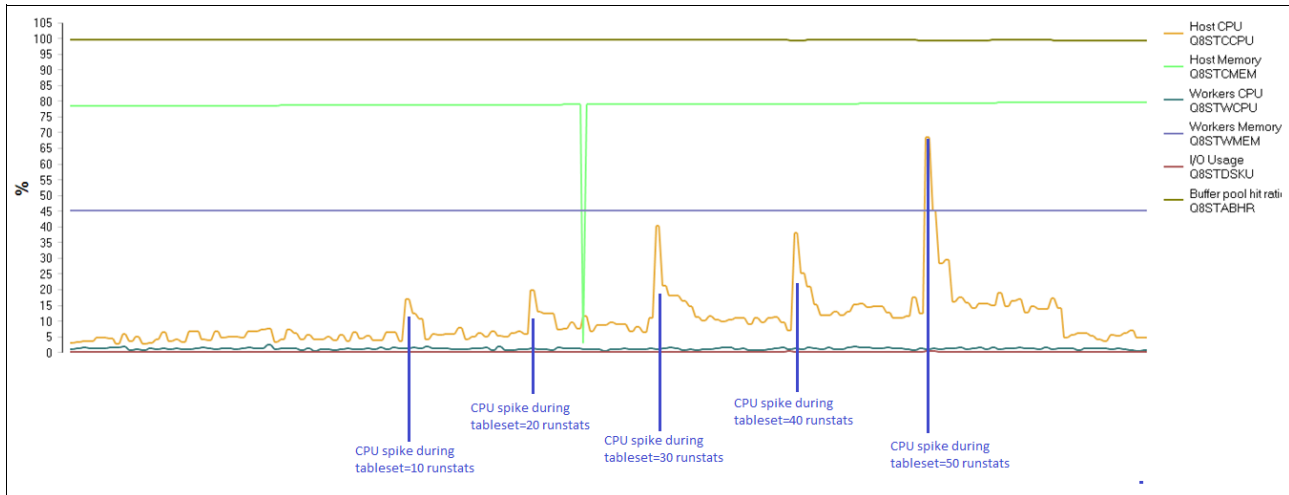


Figure 10-12 Accelerator host CPU usage during the serial explicit RUNSTATS

For the second scenario, 10 tables have explicit **RUNSTATS** running concurrently. The query workload degrades less than 4% in total elapsed time compared to the same workload without the explicit **RUNSTATS** running.

### 10.5.4 Summary

This section introduces how and when statistics are automatically triggered and collected after a table is initially loaded or fully reloaded in an accelerator. You can also explicitly run **RUNSTATS** on the accelerator if extra or specific statistics are needed.

## 10.6 Query workload comparison between Db2 12 and Db2 13 for z/OS

A comparison between Db2 12 and Db2 13 also is a part of these internal performance measurements. Considering the number of features and changes that were introduced in Db2 13 for z/OS, it is important to demonstrate that no adverse effects are observed on the accelerator-side after migrating to a new Db2 version or release.

### 10.6.1 Performance measurement

The testing consisted of running several typical workloads by using Db2 13, and then comparing the results against the baseline measurements that were captured by using Db2 12 and IDAA for z/OS 7.5.8.

#### Measurement environment

The performance evaluation was conducted on IBM z14 hardware with six general processors; one zIIP; 80 GB of memory; and running z/OS 2.3, Db2 12 FL V12R1M506, and Db2 13 FL V13R1M501.

The accelerator hardware is a 7-node full rack Sailfish system.

## Measurement workloads

The following workloads were used during testing:

- ▶ TPC-H 1 TB and 5 TB data size, single-thread workloads
- ▶ TPC-H 1 TB and 5 TB data size, multi-threaded workloads
- ▶ Initial **LOAD** Performance workloads
- ▶ Customer FD workloads

## Results summary

The single-thread TPC-H results indicate that Db2 13 provides an improvement of approximately 12% for the 5 TB workloads while staying on-par for the 1 TB workloads. The Customer FD workloads running with Db2 13 do not regress for the same workloads compared to Db2 12.

Results for the multi-threaded TPC-H workloads on Db2 13 show that the performance was also on-par compared to the Db2 12 results.

Initial table load elapsed time was also similar to the Db2 12 results.

Overall, Db2 13 offers equivalent performance for query workloads and initial loads, and provides an improvement for workloads with larger tables.

## 10.7 Performance impact of AT-TLS enablement

To address the potential need for encrypted networks and better security, performance tests were conducted to evaluate whether enabling Transport Layer Security (TLS) would impact performance, specifically when running accelerator table loads.

### 10.7.1 AT-TLS configuration for IBM Integrated Analytics Accelerator

For more information about enabling AT-TLS for IDAA for z/OS systems, see [AT-TLS configuration](#).

### 10.7.2 Results and usage considerations

After TLS was enabled on the test systems, typical load performance tests were run, and the results were compared to the results of baseline (non-TLS) load measurements. This comparison revealed the following general findings:

- ▶ Actual load elapsed time was nearly identical.
- ▶ The CPU usage increased by 30%.

Therefore, if you are considering enabling TLS, consider the potential impact on concurrent tasks when running a TLS-enabled load.

## 10.8 Performance overview for IBM Db2 for z/OS Data Gate

IBM Db2 for z/OS Data Gate makes Db2 for z/OS data easily consumable by high-intensity, read-only applications, and provides Db2 for z/OS data and meta data with a path on the journey to the cloud.

The key synchronization technology that IBM Db2 for z/OS Data Gate uses is IBM Integrated Synchronization, which is described in 10.1, “IBM Integrated Synchronization” on page 308.

IBM Db2 for z/OS Data Gate 2.1 includes many enhancements and features that can improve performance.

IBM Db2 for z/OS Data Gate 2.1 and IDAA 7.5.8 share and use the same set of stored procedures so that you can more easily maintain environments that include both products. This synergy also ensures consolidated optimized performance on the stored procedures side.

### 10.8.1 IBM Db2 for z/OS Data Gate deployment options

IBM Cloud Pak® for Data runs on Red Hat OpenShift, which enables a wide range of target platforms. You can deploy IBM Db2 for z/OS Data Gate and target databases on the following types of environments:

- ▶ An on-premises, private cloud cluster that includes Linux on IBM zSystems
- ▶ Any public cloud infrastructure that supports Red Hat OpenShift, including IBM Cloud®

Although IBM Db2 for z/OS Data Gate and IDAA share some technologies, IBM Db2 for z/OS Data Gate has its own set of typical use cases, which are illustrated in Figure 10-13. When you use IBM Db2 for z/OS Data Gate, data is synchronized from Db2 for z/OS data sources to target databases on IBM Cloud Pak for Data. For more information, see [Overview of IBM Cloud Pak for Data](#).

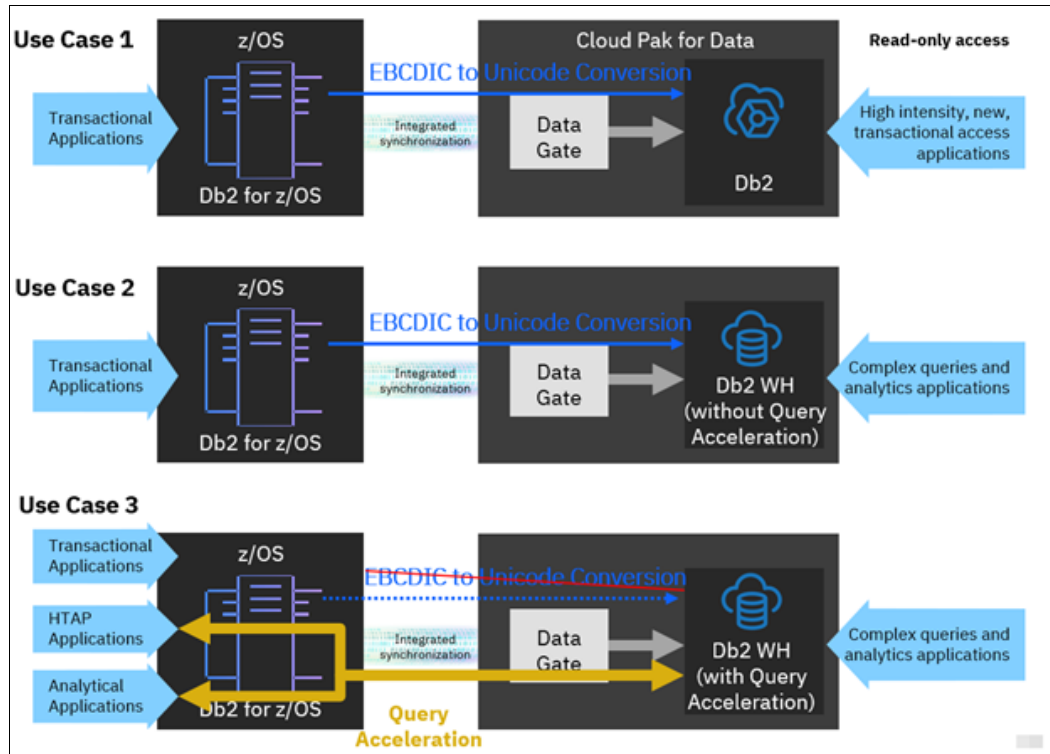


Figure 10-13 IBM Db2 for z/OS Data Gate use cases

The following sections provide more details about each type of use case.

### Use case 1: New cloud-native transactional access operations

The Db2 for z/OS data is synchronized to a target Db2 database (row-stored tables) on cloud. Cloud-native applications access the Db2 database directly by using read-only transactional operations. The source data encoding is converted from Extended Binary Coded Decimal Interchange Code (EBCDIC) to Unicode during the synchronization process.

### Use case 2: New cloud-native analytical access operations

The Db2 for z/OS data is synchronized to a target Db2 Warehouse database (column-stored tables) on the cloud. Cloud-native applications access the Db2 Warehouse database directly by using read-only analytical operations. The source data encoding is converted from EBCDIC to Unicode during the synchronization process.

### Use case 3: Query acceleration

IBM Db2 for z/OS Data Gate allows the running of analytical queries that are submitted through Db2 for z/OS to run on Db2 Warehouse on IBM Cloud Pak for Data. This use case is similar to the IDAA use case except that the target Db2 Warehouse database is on a cloud platform. No data encoding conversion happens (the target Db2 Warehouse database still stores EBCDIC data). Unlike IDAA, applications are not restricted from accessing the target Db2 Warehouse database directly. However, you must pay attention to the collation sequence, which can cause different query results.

IBM Db2 for z/OS Data Gate provides a feature that is called *retrieving a transactionally consistent result*. With this feature, even as data changes on Db2 for z/OS, you can run the query on the cloud platform and you get the same results as if the query was run on Db2 for z/OS. For more information about this feature, see [Making queries wait for Db2 Data Gate synchronization updates \(WAITFOR DATA\)](#).

## 10.8.2 Performance test strategies and results

IBM Db2 for z/OS Data Gate performance evaluations use different workloads according to these use case types. The main performance metrics include load throughput, synchronization latency, and query elapsed time.

## 10.8.3 Performance measurement: Load throughput and synchronization latency

The following environment is used for these tests.

### Measurement environment

The Db2 for z/OS side uses the following setup:

- ▶ IBM z13 hardware with eight general processors and eight zIIPs
- ▶ 750 GB of memory
- ▶ A 20 TB direct access storage device (DASD)
- ▶ 10 Gbps network
- ▶ z/OS V2R2
- ▶ Db2 12 for z/OS

The Red Hat OpenShift Cluster uses the following setup:

- ▶ One Bastion node + three master nodes (all are virtual machines): four vCPUs and 8 GB of memory
- ▶ Three worker nodes on a dedicated physical machine
- ▶ CPU: Intel 2.50 GHz x 24 physical cores (48 threads) for each worker node
- ▶ Memory: 64 GB, 256 GB, or 64 GB
- ▶ Storage: SSD 500 GB x 6 for each worker node
- ▶ 10 Gbps network

The following IBM Db2 for z/OS Data Gate, Db2, and Db2 Warehouse instances resource allocation on IBM Cloud Pak for Data were used for each of the use cases:

- ▶ Transactional (use case 1): IBM Db2 for z/OS Data Gate instance (10 vCPU and 32 GB of memory); Db2 instance (20 vCPU and 192 GB)
- ▶ Analytical (use case 2): IBM Db2 for z/OS Data Gate instance (10 vCPU and 16 GB of memory); Db2 Warehouse instance (20 vCPU and 128 GB of memory)

## Workload definitions

The following table and workload setup was used:

<b>Load</b>	Modify the TPCH.LINEITEMS table definition. Change the length of column L_COMMENT to VARCHAR (1044). Define 100 range partitions on Db2 for z/OS. Populate the source table with a total of 1 TB of data.
<b>Synchronization</b>	Define seven tables that contain many CHAR and VARCHAR columns. Duplicate these seven tables to 70 tables with different schema names. The largest table contains a total of 44 columns and 28 character columns. There are no “nullable” key columns.
<b>Application</b>	One JDBC Type 4 driver application runs in the UNIX System Services layer on the same z/OS LPAR on which Db2 for z/OS is running. The JDBC application performs insert, update, and delete operations on Db2 for z/OS tables with a specified workload pressure. Workload pressure is defined in terms of the number of row changes per second, and the workload pressure is adjustable. There are 50 concurrent threads that submit insert, update, and delete operations with a batch size of 20 rows. The Db2 for z/OS system that is used in the measurement environment that is described above can handle up to 200 KB of row changes per second.

## Results

The results for the load throughput are shown in Figure 10-14.

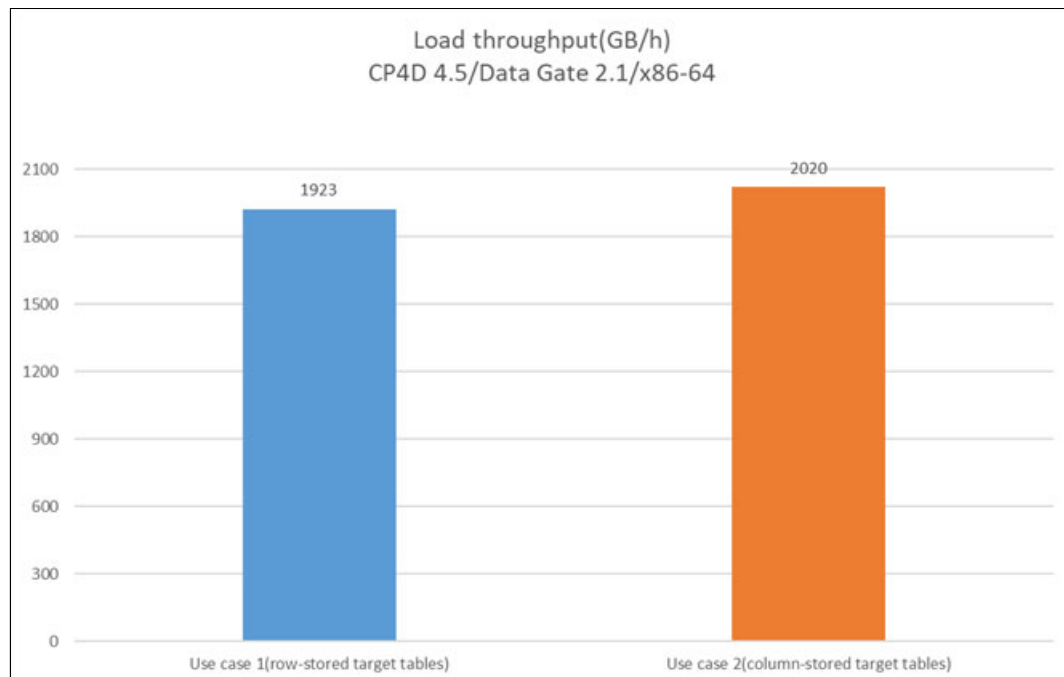


Figure 10-14 Load throughput

The synchronization latency results for different workload pressures are shown in Figure 10-15 on page 327.

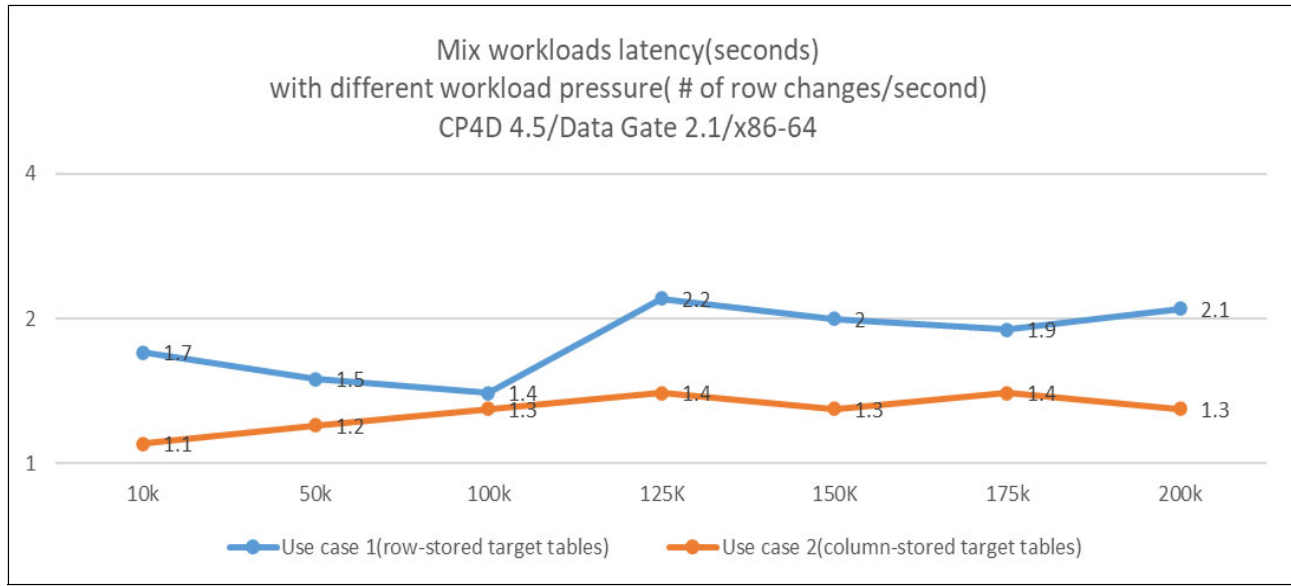


Figure 10-15 Synchronization latency for different workload pressure settings

The Mix workloads in Figure 10-15 include insert, update, and delete operations.

## Results summary

Testing revealed the following key results:

- ▶ Load achieves a 2 TB per hour throughput rate. IBM Db2 for z/OS Data Gate uses different technologies to optimize load throughput. For Db2, NOT LOGGED INITIALLY (NLI) and ADAPTIVE compression are used. With NLI, any changes that are made on a table (including insert, delete, update, or create index operations) in the same unit of work that creates the table are not logged. ADAPTIVE compression, which incorporates classic row compression and also works on a page-by-page basis to further compress data, offers the most opportunities for storage savings. Both technologies reduce transaction log-writing bottlenecks. For Db2 Warehouse, reduced logging and the usage of column-stored tables also help to avoid I/O bottlenecks.
- ▶ Network bandwidth is a key factor for load performance. The test is performed by using a 10 Gbps network. You can expect better results with higher network bandwidth.
- ▶ The synchronization test results show that IBM Db2 for z/OS Data Gate can maintain a seconds-level latency, even with a 200 KB row changes per second pressure. The record length and compression ratio are two key factors for synchronization throughput, especially for the Db2 target. The test ran with rows with a maximum record length of approximately 330 bytes. If your table record length is longer, you might see lower synchronization throughput.
- ▶ The underlying storage affects performance. Unlike with IDAA, you can choose the storage type for the Db2 and Db2 Warehouse instance on IBM Cloud Pak for Data. Using HostPath is a best practice for best performance. You can choose other storage types, such as Red Hat OpenShift Data Foundation (ODF) or IBM Spectrum® Scale, if your workload pressure is not high.
- ▶ The test results were generated from an IBM Cloud Pak for Data x86-64 platform. Tests were also run with Linux on IBM Z. The Linux on IBM Z results are on-par with the x86-64 platform but use less CPU resources.

## 10.8.4 Performance measurement: Query

To conduct the query performance measurements, the following setup is used for use case 3 (Query acceleration).

### Measurement environment

A comparable test environment with IDAA 7.5.8 was used for the query performance testing.

The Db2 for z/OS side uses the following setup:

- ▶ IBM z14 hardware with six general processors and one zIIP
- ▶ 80 GB of memory
- ▶ z/OS 2.2
- ▶ Db2 12

Db2 Warehouse message processing program (MPP) resource allocation is as follows:

- ▶ One hundred sixty-eight physical Intel cores and 2.1 TB of memory
- ▶ 44 TB of ODF storage with a 10 Gbps network

The IDAA performance test was run by using one 7-node full rack IAS, with 168 physical cores, 3.5 TB of memory, and SAN storage with a 40 Gb FC network.

### Workload definitions

Query tests use the same workloads as the IDAA for z/OS 7.5.8 performance tests.

### Results

When comparable resources are allocated for the Db2 Warehouse MPP instance, IBM Db2 for z/OS Data Gate query performance is on-par with IDAA for z/OS 7.5.8, as shown in Figure 10-16.

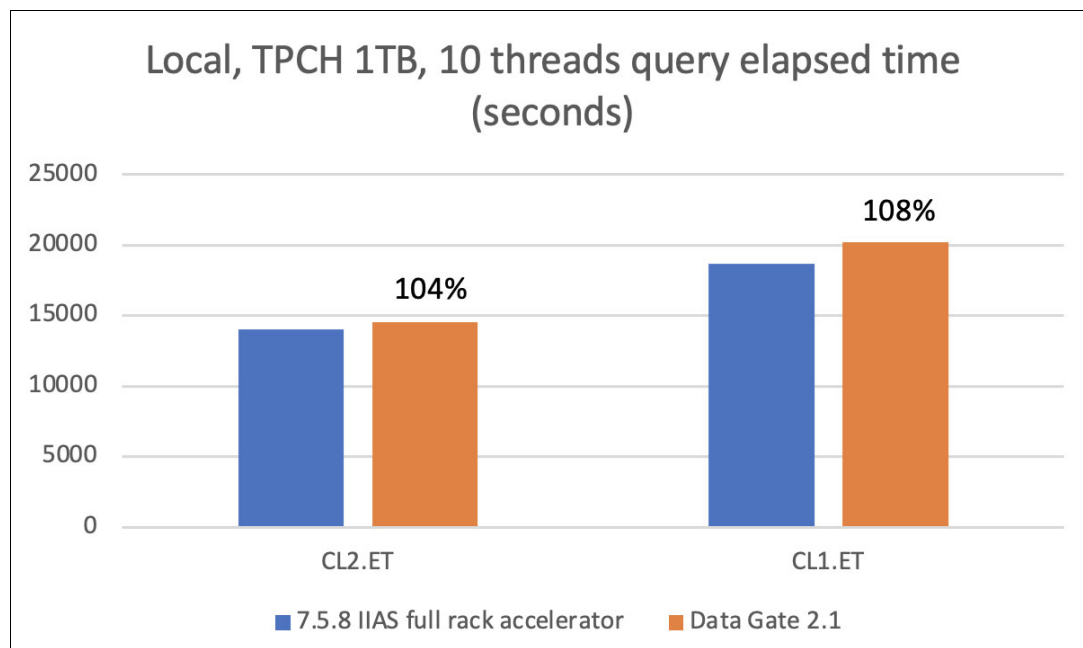


Figure 10-16 TPCH 1 TB query workload ET: IDAA for z/OS versus IBM Db2 for z/OS Data Gate



## 10.8.5 Conclusion

IBM Db2 for z/OS Data Gate provides high loading throughput, excellent latency for synchronizing data, and excellent query performance. Additionally, the log capture processing is fully zIIP enabled to ensure minimal impact to general processing on IBM Z.





# Installation and migration

This chapter describes performance-related changes to the IBM Db2 13 for z/OS installation and migration process. Although the Db2 13 new installation process remains relatively unchanged from the previous version, changes to the migration process can reduce the amount of time that it takes to migrate to Db2 13 and improve your migration experience.

Furthermore, some performance-related subsystem parameters were modified along with the various performance-related enhancements that were introduced in Db2 13.

This chapter includes the following topics:

- ▶ Migration performance
- ▶ Performance-related subsystem parameter changes

## 11.1 Migration performance

Starting with Db2 12, multiple migration modes, such as conversion mode (CM), star mode (CM\*), enabling new function mode (ENFM), and new function mode (NFM) that existed in Db2 11, were eliminated. Db2 12 introduced the concept of *function levels* (FLs), which made it possible to implement a continuous delivery model and manage groups of new enhancements. The usage of FLs starts when you begin the migration to Db2 12.

When you migrate to Db2 12 from Db2 11, the single migration job **DSNTIJTC (CATMAINT)** performs all the changes that are necessary to tailor the Db2 11 catalog and directory objects. After you run **DSNTIJTC**, the Db2 catalog reaches the catalog level of V12R1M500. Catalog and directory objects at the V12R1M500 level can be used by Db2 12 (regardless of the FL) and by Db2 11 with the fall-back SPE applied. As a result, if the fall-back SPE is in place, falling back to Db2 11 or coexisting with Db2 11 members in a data-sharing configuration is possible after the catalog changes to the Db2 12 format.

In Db2 12, you do not run a job to enable a new function. Instead, you use the new Db2 **ACTIVATE** command to activate a new function. Because only one step is needed to update the catalog during migration, catalog availability is improved. As a result, applications that access the catalog can expect less interruption during the migration.

Although there are no changes to the installation process in Db2 13 when installing a new Db2 subsystem, the migration process from Db2 12 to Db2 13 changed to provide performance improvements and a better user experience by eliminating catalog and directory updates during the migration. Eliminating catalog and directory updates was achieved by requiring Db2 12 to be updated and activated at FL V12R1M510 (FL 510) before migrating to Db2 13. When you activate FL 510 in Db2 12, Db2 validates the necessary catalog level and structure updates. Db2 13 still uses the initial release migration job **DSNTIJTC (CATMAINT)**, but in Db2 13 **CATMAINT** no longer changes the structure of the Db2 catalog. It does not create any tables or indexes, and it does not add columns to existing catalog tables. This initial **CATMAINT** process sets internal information to indicate that the catalog level is now V13R1M100 and that the FL is V13R1M100 (FL 100). Unlike in Db2 12, **CATMAINT** functions as a switch to indicate migration completion without modifying any catalog or directory objects, which speed up the migration process and improves subsystem availability.

After migrating a Db2 subsystem to Db2 13 FL 100 and running it with an expected level of stability for a period, you can activate FL V13R1M500 (FL 500). Activating FL 500 also does not result in any catalog changes. However, after you activate FL 500, falling back to Db2 12 or release coexistence in a data-sharing environment is no longer possible. Release coexistence between Db2 12 and Db2 13 is available only while FL 100 is the highest activated FL in Db2 13.

### 11.1.1 Requirements

Db2 13 requires the following hardware and software:

- ▶ IBM zEC12 or later
- ▶ z/OS 2.4 or later
- ▶ Db2 12 with FL 510 and with fall-back SPE (PH37108) applied

## 11.1.2 Performance measurements

The following performance measurements were collected in a controlled lab setting to demonstrate the relative performance of migrating to different versions of Db2. These measurements reflect the results of migrating from Db2 10 NFM to Db2 13 at FL V13R1M500 (FL 500). Measurements are provided for all the steps that are required to migrate from a Db2 10 NFM system to a Db2 13 FL 500 system. These measurements include the CPU and elapsed time to migrate from one version to the next version. The specifics of the performance of catalog update and FL activation from Db2 13 FL 500 to FL 501 also are provided.

**Note:** These measurements are to be used only as a reference to estimate elapsed time and CPU time of the migration jobs when migrating from Db2 10 to Db2 13. There is no direct migration path from Db2 10 to Db2 13.

### Measurement environment

The migration performance measurements were conducted by using the following environment:

- ▶ z15
- ▶ z/OS 2.5
- ▶ Four general-purpose central processors (GPCs)
- ▶ Db2 one-way data sharing
- ▶ A Db2 catalog size of 343 MB in Db2 10

### Results

Elapsed time and CPU time of the migration jobs are provided for each version migration of a Db2 subsystem migrating from Db2 10 to Db2 13:

- ▶ Db2 10 NFM migration to Db2 11 NFM
- ▶ Db2 11 NFM migration to Db2 12 FL 500
- ▶ Db2 12 FL 500 mid-release migration to Db2 12 FL 510
- ▶ Db2 12 FL 510 migration to Db2 13 FL 500
- ▶ Db2 13 FL 500 migration to Db2 13 FL 501

### **Db2 10 NFM to Db2 11 NFM migration**

Table 11-1 provides the elapsed time and CPU usage for the following migration jobs:

- ▶ Migration from Db2 10 NFM to Db2 11 CM:  
**DSNTIJTC: CATMAINT UPDATE**
- ▶ Migration from Db2 11 CM to Db2 11 ENFM:  
**DSNTIJEN: Online REORG** of the following tables:
  - SYSLGRNX
  - SYSCOPY
  - SYSRTSTS
  - SYSTSIXS
  - SYSSTR
  - SYSTSTAB
- ▶ Migration from Db2 11 ENFM to Db2 11 NFM:  
**DSNTIJNF: Turn on NFM.**

*Table 11-1 Elapsed and CPU time of the migration steps from Db2 10 NFM to Db2 11 NFM*

<b>Migration steps</b>	<b>Elapsed time (seconds)</b>	<b>CPU time (seconds)</b>
Db2 10 NFM to Db2 11 CM <b>DSNTIJTC (CATMAINT UPDATE)</b>	7.36	0.452
Db2 11 CM to Db2 11 ENFM <b>(DSNTIJEN)</b>	45.684	2.009
Db2 11 ENFM to Db2 11 NFM <b>(DSNTIJNF)</b>	1.956	0.005
Total	55.001	2.466

The results show that ENFM uses the most CPU and takes longer to complete, which is a result of the online **REORG** of the table spaces.

### **Db2 11 NFM to Db2 12 FL 500 migration**

Table 11-2 provides the elapsed time and CPU usage for the following migration jobs:

- ▶ Migration from Db2 11 NFM to Db2 12 FL 100:  
**DSNTIJTC: CATMAINT UPDATE V12R1M500**
- ▶ Activation of Db2 12 FL 500:  
**DSNTIJAF: ACTIVATE FUNCTION LEVEL (V12R1M500)**

*Table 11-2 Elapsed and CPU time of the migration from Db2 11 NFM to Db2 12 FL 500*

<b>Migration steps</b>	<b>Elapsed time (seconds)</b>	<b>CPU time (seconds)</b>
Db2 11 NFM to Db2 12 FL 100 <b>DSNTIJTC (CATMAINT UPDATE)</b>	9.703	0.388
Activate Db2 12 FL 500 <b>(DSNTIJAF)</b>	1.315	0.015
Total	11.018	0.403

The migration step that performs the **CATMAINT UPDATE** takes longer to complete and uses the most CPU because of the catalog changes that it must perform.

**Db2 12 FL 500 to Db2 12 FL 510 mid-release migration**

Table 11-3 provides the elapsed time and CPU usage for the following migration jobs:

- ▶ Migration from Db2 12 FL 500 to Db2 12 FL 510:  
**DSNTIJTC: CATMAINT UPDATE V12R1M510**
- ▶ Activation of Db2 12 FL 510  
**DSNTIJAF: ACTIVATE FUNCTION LEVEL (V12R1M510)**

*Table 11-3 Elapsed and CPU time of the migration from Db2 12 FL 500 to Db2 12 FL 510*

Migration steps	Elapsed time (seconds)	CPU time (seconds)
Db2 12 FL 500 to Db2 12 FL 510 <b>DSNTIJTC (CATMAINT UPDATE)</b>	1.574	0.046
Activate Db2 12 FL 510 <b>(DSNTIJAF)</b>	1.671	0.013
Total	3.244	0.059

Mid-release migration from Db2 V12R1M500 to Db2 12 FL 510 takes 3.244 seconds to complete and consumes 0.59 CPU seconds. During **CATMAINT UPDATE** from FL 500 to FL 510, catalog changes are applied for FL 502, FL 503, FL 505, FL 507, and FL 509.

**Db2 12 FL 510 to Db2 13 FL 500 migration**

Table 11-4 provides the elapsed time and CPU usage for the following migration jobs:

- ▶ Migration from Db2 V12R1M510 to Db2 13 FL 100:  
**DSNTIJTC: CATMAINT UPDATE V13R1M100**
- ▶ Activation of Db2 13 FL 500:  
**DSNTIJAF: ACTIVATE FUNCTION LEVEL (V13R1M500)**

*Table 11-4 Elapsed and CPU time of the migration from Db2 12 FL 510 to Db2 13 FL 500*

Migration steps	Elapsed time (seconds)	CPU time (seconds)
Db2 12 FL 510 to Db2 13 FL 100 <b>DSNTIJTC (CATMAINT UPDATE)</b>	2.111	0.016
Activate Db2 13 FL 500 <b>(DSNTIJAF)</b>	2.191	0.016
Total	4.302	0.032

Migration from Db2 12 FL 510 to Db2 13 FL 100 completes quickly with hardly any CPU consumption compared to the migration from Db2 11 NFM to Db2 12 FL 500 because no catalog changes are made during this migration phase. Unlike Db2 12, the **DSNTIJTC** job performs a **CATMAINT UPDATE** to FL 100 in Db2 13 and not to FL 500.

### Db2 13 FL 500 to Db2 13 FL 501

Table 11-5 provides elapsed time and CPU usage for the following migration jobs:

- ▶ Migration from Db2 13 FL 500 to Db2 13 FL 501:  
**DSNTIJTC: CATMAINT UPDATE V13R1M501**
- ▶ Activation of Db2 13 FL 501:  
**DSNTIJAF: ACTIVATE FUNCTION LEVEL(V13R1M501)**

Table 11-5 Elapsed and CPU time of the migration from Db2 13 FL 500 to Db2 13 FL 501

Migration steps	Elapsed time (seconds)	CPU time (seconds)
Db2 13 FL 500 to Db2 13 FL 501 <b>DSNTIJTC (CATMAINT UPDATE)</b>	1.662	0.117
Activate Db2 13 FL 501 <b>(DSNTIJAF)</b>	1.513	0.012
Total	3.175	0.129

A mid-release migration from Db2 V13R1M500 to Db2 V13R13M501 takes 3.175 seconds to complete and consumes 129 milliseconds of CPU usage. This phase involves changes to the Db2 catalog.

### Summary of results

The evaluation and comparison were done by using a small Db2 subsystem to illustrate the relative performance difference between the migration of different Db2 versions. The migration of a large Db2 subsystem with several databases and packages takes more elapsed time and CPU time (for more information, see *IBM Db2 12 for z/OS Performance Topics*, SG24-8404).

Comparing the different migrations from Db2 10 NFM to Db2 13 FL 500 shows that the overall elapsed time and CPU usage improved for migrations to Db2 12 and Db2 13 when compared to the previous version.

Migration from Db2 11 NFM to Db2 V12R1M500 completes 79% faster and uses 83% less CPU when compared to migration from Db2 10 NFM to Db2 11 NFM.

Migration from Db2 V12R1M510 to Db2 V13R1M500 completes 61% faster and uses 92% less CPU when compared to migration from Db2 11 NFM to Db2 V12R1M500.

## 11.1.3 Monitoring information

The performance of the migration jobs was measured by using RMF Workload Manager (WLM) reporting.

You can query the SYSLEVELUPDATES catalog table to see the history of catalog changes and FL activations.



## 11.1.4 Usage considerations

For a successful migration from Db2 11 to Db2 12, convert bootstrap data sets (BSDS) to extended format by using the **DSNTIJCB** sample job. As a best practice, convert Db2 catalog and directory objects to extended RBA/LRSN format. Catalog and directory objects that go beyond 6 bytes can cause a serious problem in which you can no longer convert any page sets because **REORG** cannot update the catalog. This step takes some time to complete compared to all the other steps during migration. For example, during this migration exercise in our lab environment, this step took 470 seconds to complete and used 16 CPU seconds.

Before you migrate from Db2 12 to Db2 13, you must complete the following steps:

1. Db2 12 must be activated at FL 510.
2. Before Db2 12 can be successfully activated at FL 510, packages that were bound earlier than Db2 11 and were active in the last 18 months must be rebound.

**Note:** The rebinding of SQL PL stored procedure rebinds only the SQL statements and not the control statements. Therefore, regenerate the SQL PL stored procedure by using the **ALTER PROCEDURE... REGENERATE** command. Run this command for SQL PL procedures that are generated before Db2 11 to avoid auto bind in Db2 13. For more information, see [Rebind old plans and packages in Db2 12 to avoid disruptive autobinds in Db2 13](#).

## 11.1.5 Conclusion

As demonstrated by the measurements, the Db2 migration process has made great strides in improving both the user experience and performance since Db2 11. The Db2 13 migration took another leap in improving the performance of the migration process by eliminating changes to the catalog structure during the **CATMAINT** migration step.

## 11.2 Performance-related subsystem parameter changes

To support the many new features and enhancements that Db2 13 introduces, some new Db2 subsystem parameters were added, some default values of existing parameters were changed, and some parameters were made obsolete. The subsystem parameter changes that are related to Db2 performance are described in this section.

### 11.2.1 New subsystem parameters that are related to performance

The following new subsystem parameters can affect Db2 performance.

#### **UTILITY\_HISTORY**

The **UTILITY\_HISTORY** subsystem parameter specifies whether utility history information is collected. This subsystem parameter is ignored in FL V13R1M500 and earlier.

In FL V13R1M501 or later, the following values for **UTILITY\_HISTORY** are acceptable:

- ▶ **NONE:** Utility information is not collected. Information is not inserted into the **SYSIBM.SYSUTILITIES** catalog table. **NONE** is the default value.
- ▶ **UTILITY:** A **SYSIBM.SYSUTILITIES** row is inserted at the beginning of each utility run. The information in the row is updated as the utility progresses. When a utility completes, the final information is stored in the row.

This subsystem parameter is changeable online. All members of a data-sharing group should have the same setting or the utility history will be incomplete.

For more information about performance measurement data with the new **UTILITY\_HISTORY** subsystem parameter, see Chapter 9, “IBM Db2 for z/OS utilities” on page 279.

### **SPREG\_LOCK\_TIMEOUT\_MAX**

The **SPREG\_LOCK\_TIMEOUT\_MAX** subsystem parameter controls the maximum value that can be specified for the **SET CURRENT LOCK TIMEOUT** statement, and whether -1 can be specified for the special register to indicate that there is no limit for how long an application can wait for a lock. If -1 is specified, any valid value for the **SET CURRENT LOCK TIMEOUT** statement can be specified.

In FL V13R1M100 and later, the following options are valid:

- ▶ -1: Any supported value can be specified in a **SET CURRENT LOCK TIMEOUT** statement. -1 is the default value.
- ▶ 0 - 32767: The range of values that can be specified in a **SET CURRENT LOCK TIMEOUT** statement.

## **11.2.2 Performance-related subsystem parameters that were updated**

The following subsystem parameters were updated in Db2 13 and can affect performance.

### **DSMAX**

The **DSMAX** subsystem parameter specifies the maximum number data sets that can be open at one time by a Db2 subsystem. The highest possible value of this parameter is increased from 200000 to 400000.

For more information about the performance tests that were conducted to measure the effect of changes to **DSMAX**, see Chapter 2, “Scalability and availability” on page 13.

### **FTB\_NON\_UNIQUE\_INDEX**

The **FTB\_NON\_UNIQUE\_INDEX** subsystem parameter specifies whether fast index traversal (fast traversal blocks (FTB)) is enabled for non-unique indexes.

The default value of **FTB\_NON\_UNIQUE\_INDEX** is changed from NO to YES in Db2 13, which means that non-unique indexes are automatically eligible for FTB processing if they meet other eligibility criteria, such as the maximum key length.

You might experience an increase in internal resource lock manager (IRLM) CPU time for data-sharing systems because more indexes are likely to be using FTB after this subsystem parameter change enables non-unique indexes for FTB processing. However, this increase should be offset by a decrease in transaction class 2 CPU time from using FTB processing for non-unique indexes. For more information about the actual measurement numbers, see Chapter 2, “Scalability and availability” on page 13.

### **PAGESET\_PAGENUM**

The **PAGESET\_PAGENUM** subsystem parameter specifies whether partition-by-range (PBR) table spaces and associated partitioned indexes (PIs) are created to use absolute page numbering (APN) across partitions or relative page numbering (RPN).

The default setting of **PAGESET\_PAGENUM** is changed from ABSOLUTE to RELATIVE in Db2 13. As a best practice, all members of a data-sharing group should use the same value.

RPN was introduced in Db2 12. The default **PAGESET\_PAGENUM** setting remained APN in Db2 12. Changing the default to RPN in Db2 13 is consistent with the new feature that allows conversion from partition-by-growth (PBG) to PBR in Db2 13, which supports conversion only to RPN, not to APN. Creating a table space as RPN, or converting directly from PBG to PBR RPN, eliminates the need for a costly conversion from APN to RPN later because such a conversion requires a **REORG** of the entire table space.

For more information about the performance measurements of converting from PBG to PBR, see Chapter 9, “IBM Db2 for z/OS utilities” on page 279.

## **STATIME\_MAIN**

The **STATIME\_MAIN** subsystem parameter specifies the time interval, in seconds, for the collection of those interval-driven statistics that are not collected at the interval that is specified by the **STATIME** subsystem parameter.

In Db2 13, the default setting of **STATIME\_MAIN** is changed from 60 seconds to 10 seconds. As a best practice, all members of a data-sharing group should use the same **STATIME\_MAIN** value.

The Db2 12 default **STATIME\_MAIN** statistics interval setting of 60 seconds makes it difficult for database administrators and Db2 system programmers to identify true workload peaks by using Db2 statistics for subsystem level performance tuning and planning. Similarly, when slowdowns in the range of 5 - 15 seconds occur, diagnosing the problem becomes difficult by using 60-second-interval data. Using a more granular statistics collection default of 10 seconds helps to speed up performance diagnosis.

## **OUTBUFF**

The **OUTBUFF** subsystem parameter specifies the size of the output log buffer, in kilobytes, for writing active log data sets.

The current default of 4000 does not reflect current best practices. The default was changed to 102400 (equivalent to 100 MB). Having enough **OUTBUFF** is critical for update (I/U/D) performance.

If you are using an **OUTBUFF** size that is greater than 100 MB, you do not need to consider updating your definition.

## **SRTPOOL**

The **SRTPOOL** subsystem parameter specifies the maximum amount of storage, in kilobytes, that is allowed for the Relational Data System (RDS) sort pool (for an individual sort).

The default value was changed to from 10000 to 20000 to reflect best practices to obtain CPU reduction by reducing the number of sort runs. If you are using an **SRTPOOL** size that is more than 20000, you do not need to consider updating your definition.

For more information about the performance test results that are related to the **SRTPOOL** subsystem parameter change, see Chapter 3, “Synergy with the IBM Z platform” on page 43.

## **REORG\_INDEX\_NOSYSUT1**

The **REORG\_INDEX\_NOSYSUT1** subsystem parameter specifies whether **REORG INDEX SHRLEVEL REFERENCE** or **CHANGE** uses the **NOSYSUT1** behavior.

Starting in Db2 13 FL V13R1M500, the default value of **REORG\_INDEX\_NOSYSUT1** was changed to YES, which will be the only behavior for **REORG INDEX** when you specify the **SHRLEVEL REFERENCE** or **CHANGE** keywords. Any user specification of this subsystem parameter is ignored.

Performance measurements that use **NOSYSUT1** improved in both elapsed time and CPU time. For more information about the results of these measurements, see Chapter 9, “IBM Db2 for z/OS utilities” on page 279.

### **MAXCONQN**

If the number of active in-use database access threads (DBATs) reaches the **MAXDBAT** threshold, the **MAXCONQN** subsystem parameter specifies the maximum number of inactive or new connection requests that can be queued waiting for a DBAT to process the request.

The default value was changed from OFF to ON to reflect the best practices for improving the application availability.

### **MAXCONQW**

The **MAXCONQW** subsystem parameter specifies the maximum length of time that a client connection waits for a DBAT to process the next unit-of-work or a new connection request.

The default value was changed from OFF to ON to reflect the best practices for improving the application availability.

### **EDM\_SKELETON\_POOL**

The **EDM\_SKELETON\_POOL** subsystem parameter determines the maximum size, in kilobytes, that is used for the environmental descriptor manager (EDM) skeleton pool. This storage is above the 2 GB bar.

The default value was changed from 51200 to 81920 to improve the performance of accessing Db2 plans and packages. If you are using an **EDM\_SKELETON\_POOL** size that is more than 81920, you do not need to consider updating your definition.

### **EDMDBDC**

The **EDMDBDC** subsystem parameter determines the maximum amount of EDM storage, in kilobytes, that is used for database descriptors (DBDs). This storage pool is above the 2 GB bar.

The default value was changed from 23400 to 40960 to improve the performance of accessing the DBDs. If you are using an **EDMDBDC** size that is more than 40960, you do not need to consider updating your definition.

### **MAXSORT\_IN\_MEMORY**

The **MAXSORT\_IN\_MEMORY** subsystem parameter specifies the maximum storage allocation, in kilobytes, for a query that contains an **ORDER BY** clause, a **GROUP BY** clause, or both. The storage is allocated only during the processing of the query. Increasing the value in this field can improve performance of such queries but might require a larger amount of real storage when several such queries run simultaneously.

The default value was changed from 1000 to 2000 to expand the opportunity of using in-memory sort to reduce the CPU time and work-file demand from the sort-intensive SQL statements. If you are using a **MAXSORT\_IN\_MEMORY** size that is more than 2000, you do not need to consider updating your definition.

### **NUMLKTS**

The **NUMLKTS** subsystem parameter specifies the default maximum number of page, row, large object (LOB), or XML locks that an application can hold simultaneously in a table or table space. If a single application exceeds the maximum number of locks in a single table or table space, lock escalation occurs.

The default value was changed from 2000 to 5000 to reduce the application concurrency impact from lock escalation. If you are using a **NUMLKTS** value more than 5000, you do not need to consider updating your definition.

## **NUMLKUS**

The **NUMLKUS** subsystem parameter specifies the maximum number of page, row, LOB, or XML locks that a single application can hold concurrently for all table spaces.

The default value was changed from 10000 to 20000 to reduce the application impact. If you are using a **NUMLKUS** value that is more than 20000, you do not need to consider updating your definition.

## **IRLMRWT**

The **IRLMRWT** subsystem parameter controls the number of seconds that are to elapse before a resource timeout is detected.

This subsystem parameter is changed to be online-updatable to support more granular specification of the lock timeout value.

In previous Db2 versions, the following IRLM command could be used to dynamically change the IRLM timeout value:

```
F ir1mproc,SET,TIMEOUT=xxx,subsystem-name
```

However, this approach was not ideal, and it is not recommended because using the command does not update the value of the **IRLMRWT** subsystem parameter.

## **STATPGSAMP**

The **STATPGSAMP** subsystem parameter specifies whether the **RUNSTATS** utility, or other utilities with inline statistics, use page-level sampling by default for universal table spaces (UTSs).

In Db2 FL V13R1M500 and later, **STATPGSAMP** also applies to the collection of inline statistics when the **LOAD** or **REORG TABLESPACE** utilities run with the **STATISTICS** keyword.

### **11.2.3 Obsolete subsystem parameters that are related to performance**

The following performance-related subsystem parameters are obsolete in Db2 13.

#### **REALSTORAGE\_MANAGEMENT**

Before Db2 13, the **REALSTORAGE\_MANAGEMENT** subsystem parameter was used to specify how Db2 should manage its real storage usage. In Db2 12, when **REALSTORAGE\_MANAGEMENT** is set to **AUTO**, Db2 pro-actively frees unused 64-bit real storage frames by issuing z/OS IARV64 **DISCARDATA** requests when a thread deallocates, or every 120 commits when the thread is reused.

Db2 13 uses enhanced automatic behavior, which avoids causing unnecessary z/OS Real Storage Manager (RSM) serialization from **DISCARDATA** requests, which are not available in Db2 12. Therefore, the **REALSTORAGE\_MANAGEMENT** subsystem parameter is obsolete.

If the **REALSTORAGE\_MANAGEMENT** parameter is specified in the **DSNTIJUZ** job for a Db2 13 system, you receive the following warning message:

```
REALSTORAGE_MANAGEMENT CAN NO LONGER BE SPECIFIED.
```

For more information about the performance tests that were conducted to measure the effect of this **REALSTORAGE\_MANAGEMENT** change, see Chapter 2, “Scalability and availability” on page 13.

## **AUTHCACH**

Before Db2 13, the **AUTHCACH** subsystem parameter was used to specify the size, in bytes per plan, of the authorization cache that is used if no **CACHESIZE** is specified on the **BIND PLAN** subcommand.

In Db2 13, the plan authorization cache size is 4096 by default. **AUTHCACH** was made obsolete for better authorization cache management. If you still want to change the size or eliminate the cache, you can do so at the plan level by specifying the **BIND/REBIND CACHESIZE** parameter.

For more information about the performance measurements with plan authorization cache, see Chapter 2, “Scalability and availability” on page 13.

## **PARA\_EFF**

Before Db2 13, the **PARA\_EFF** subsystem parameter was used to control the efficiency that Db2 assumes for parallelism during access path selection. For best results, the default value of 50 was recommended.

In Db2 12, the value that you specified for this parameter is acknowledged, but its use was discouraged. Support for this subsystem parameter was removed in Db2 13.



## Monitoring and instrumentation

Db2 13 for z/OS introduces many changes and enhancements to monitoring and instrumentation. These changes include the addition of new Instrumentation Facility Component Identifiers (IFCIDs) for trace records, changes to some existing IFCIDs, and updates to Db2 accounting and statistics information. Almost all these enhancements are available in Db2 13 function level (FL) 100, and many of them are available in Db2 12. When a different FL or specific APAR is required for an enhancement, that information is provided.

This chapter includes the following topics:

- ▶ New IFCID 396 to track index page split activity
- ▶ Enhanced distributed thread monitoring
- ▶ IFCID 003: Recording the longest wait time for certain suspension types
- ▶ Group buffer pool residency time enhancements
- ▶ IFCID 369 enabled by default when STATISTICS CLASS(1) starts
- ▶ IFCID changes for application timeout and deadlock control
- ▶ Partition range support for IFCID 306 READS requests
- ▶ New fields in IFCID 3 for SQL Data Insights
- ▶ New fields in IFCID 2 for plan authorization cache
- ▶ New field in IFCID 389 for fast index traversal
- ▶ IFCID changes that are related to storage manager large object contraction
- ▶ IFCID changes that are related to latch-level expansion
- ▶ Obtaining the most current information about IFCID changes

## 12.1 New IFCID 396 to track index page split activity

Db2 13 includes a new IFCID 396 trace record and three new catalog columns to trace and capture more pertinent information about index page split activity, which helps you identify and analyze the root cause of **INSERT** performance issues.

For more information about this new IFCID 396 trace record and how to use it, along with information about three new catalog fields, see Chapter 7, “Application concurrency” on page 241.

## 12.2 Enhanced distributed thread monitoring

Db2 13 includes two new trace records, IFCID 411 and 412, to collect statistics information similar to the information that is collected by IFCID 365 (detailed location statistics), but at the client application name level or client user ID level instead of by location or IP address. Existing trace records IFCID 1 and 365 also were improved.

**Note:** It is a best practice to collect and externalize only one of the DDF statistics trace records (IFCID 365, 411, or 412). You can start all three, but doing so increases the number of trace records that are written to the trace destination.

### 12.2.1 New IFCID 411 for recording DDF application statistics

IFCID 411 was introduced to record detailed statistics about the client applications that are involved in communicating with a Db2 subsystem by using the Distributed Relational Database Architecture (DRDA) protocol. IFCID 411 can be started by using statistics trace class 10.

Table 12-1 shows the IFCID 411 fields.

Table 12-1 IFCID 411 fields

Column name	Data type	Description
QLAPPN	CHAR(16)	The name of the application that is running at the remote site.
QLAPPRID	CHAR(8)	The product ID of the remote location from which the remote application connects.
QLAPCOMR	BIGINT	The number of commit requests that are received from the requester (single-phase commit protocol), and the number of committed requests that are received from the coordinator (two-phase commit protocol).
QLAPABRR	BIGINT	The number of abort requests that are received from the requester (single-phase commit protocol), and the number of back-out requests that are received from the coordinator (two-phase commit protocol).
QLAPRLV	CHAR(16)	The product level, if known.
QLAPNREST	BIGINT	The number of times that the application reported a connection or application condition from a REST service request.
QLAPNSSR	BIGINT	The number of times that the application reported a connection or application condition from setting a special register through a profile.



Column name	Data type	Description
QLAPNSGV	BIGINT	The number of times that the application reported a connection or application condition from setting a global variable through a profile.
QLAPHCRSR	BIGINT	The number of times that the application used a cursor that was defined as WITH HOLD and was not closed. That condition prevented Db2 from pooling database access threads (DBATs).
QLAPDGTT	BIGINT	The number of times that the application did not drop a declared temporary table. That condition prevented Db2 from pooling DBATs.
QLAPKPDYN	BIGINT	The number of times that the application used a <b>KEEPDYNAMIC</b> package. That condition prevented Db2 from pooling DBATs.
QLAPHIPRF	BIGINT	The number of times that the application used a high-performance DBAT. That condition prevented Db2 from pooling DBATs.
QLAPHLOBLOC	BIGINT	The number of times that the application had a held large object (LOB) locator. That condition prevented Db2 from pooling DBATs.
QLAPSPCMT	BIGINT	The number of times that a <b>COMMIT</b> was issued in a stored procedure. That condition prevented Db2 from pooling DBATs.
QLAPNTHDPQ	BIGINT	The number of times that a thread that was used by a connection from the application was queued because a profile exception threshold was exceeded.
QLAPNTHDPT	BIGINT	The number of times that a thread that was used by a connection from the application was terminated because a profile exception threshold was exceeded.
QLAPNTHDA	BIGINT	The number of times that a thread that was used by a connection from the application abended.
QLAPNTHDC	BIGINT	The number of times that a thread that was used by a connection from the application was canceled.
QLAPNTHD	INTEGER	The current number of active threads for the application.
QLAPHTHD	INTEGER	For a statistics trace, the highest number of active threads during the current statistics interval; for a <b>READS</b> request, the highest number of active threads since DDF was started.
QLAPTHDTM	BIGINT	The number of threads that were queued because the <b>MAXDBAT</b> subsystem parameter value was exceeded.
QLAPTHDTI	BIGINT	The number of threads that were terminated because the <b>IDTHTOIN</b> subsystem parameter value was exceeded.
QLAPTHDTC	BIGINT	The number of threads that were terminated because the <b>CANCEL THREAD</b> command was issued.
QLAPTHDTR	BIGINT	The number of threads that were terminated because a profile exception condition for idle threads was exceeded.
QLAPTHDTK	BIGINT	The number of threads that were terminated because the threads were running under <b>KEEPDYNAMIC</b> refresh rules, and the idle time exceeded the <b>KEEPDYNAMICREFRESH</b> idle time limit (20 minutes).
QLAPTHDTF	BIGINT	The number of threads that were terminated because the threads were running under <b>KEEPDYNAMIC</b> refresh rules, and the time that the threads were in use exceeded the <b>KEEPDYNAMICREFRESH</b> in-use time limit.
QLAPTHDTN	BIGINT	The number of threads that were terminated due to network termination.

The information in IFCID 411 can help you identify applications that are using resources inefficiently in the following types of situations:

- ▶ An application is not releasing a connection in a timely manner.
- ▶ An application is causing unexpected thread termination.
- ▶ An application is monitored by a profile, but thresholds must be adjusted (they are too high or too low).
- ▶ An application is monopolizing resources (such as opening multiple threads) and should be monitored.

If the trace identifies more than 50 applications to be reported, Db2 generates multiple IFCID 411 records. For Instrumentation Facility Interface (IFI) **READS** requests, only those IFCID 411 records that fit into the buffer are returned. The interval at which the IFCID 411 statistics trace records are produced is controlled by the **STATIME\_MAIN** subsystem parameter.

These counters are maintained for a server site only at the client application level. When IFCID 411 is written as part of a statistics trace, the high-water mark (HWM) is the maximum value that is observed since the last statistics trace interval. When IFCID 411 is created for a **READS** request, the HWM is the maximum value that is observed since DDF was started.

IFCID 411 information is also available in Db2 12 for z/OS after applying the PTF for PH40244.

## 12.2.2 New IFCID 412 for recording DDF client user ID statistics

IFCID 412 was introduced to record detailed statistics about the client user IDs that are involved in communicating with Db2 subsystems by using the DRDA protocol. IFCID 412 can be started by using statistics trace class 11.

Table 12-2 shows the IFCID 412 fields.

Table 12-2 IFCID 412 fields

Column name	Data type	Description
QLAUUSRI	CHAR(16)	The name of the client user ID under which the connection from the remote application to the local site is established.
QLAUPRID	CHAR(8)	The product ID of the remote application from which the application connects.
QLAUCOMR	BIGINT	The number of commit requests that were received from the requester (single-phase commit protocol), and the number of committed requests that were received from the coordinator (two-phase commit protocol).
QLAUABRR	BIGINT	The number of abort requests that were received from the requester (single-phase commit protocol), and the number of back-out requests that were received from the coordinator (two-phase commit protocol).
QLAUPRLV	CHAR(16)	The product level, if known.
QLAUNREST	BIGINT	The number of times that an application that is run by the specified client user ID reported a connection or application condition from a REST service request.
QLAUNSSR	BIGINT	The number of times that an application that is run by the specified client user ID reported a connection or application condition from setting a special register through a profile.

Column name	Data type	Description
QLAUNSGV	BIGINT	The number of times that an application that is run by the specified client user ID reported a connection or application condition from setting a global variable through a profile.
QLAUHCRSR	BIGINT	The number of times that an application that is run by the specified client user ID used a cursor that was defined as <b>WITH HOLD</b> and was not closed. That condition prevented Db2 from pooling DBATs.
QLAUDGTT	BIGINT	The number of times that an application that is run by the specified client user ID did not drop a declared temporary table. That condition prevented Db2 from pooling DBATs.
QLAUKPDYN	BIGINT	The number of times that an application that is run by the specified client user ID used <b>KEEPDYNAMIC</b> packages. That condition prevented Db2 from pooling DBATs.
QLAUHIPRF	BIGINT	The number of times that an application that is run by the specified client user ID used a high-performance DBAT. That condition prevented Db2 from pooling DBATs.
QLAUHLOBLOC	BIGINT	The number of times that an application that is run by the specified client user ID used a held LOB locator. That condition prevented Db2 from pooling DBATs.
QLAUSPCMT	BIGINT	The number of times that a <b>COMMIT</b> was issued in a stored procedure that was called by the specified client user ID. That condition prevented Db2 from pooling DBATs.
QLAUNTHDPQ	BIGINT	The number of times that a thread that was used by a connection from the application that is run by the specified client user ID was queued because a profile exception threshold was exceeded.
QLAUNTHDPT	BIGINT	The number of times that a thread that was used by a connection from the application that is run by the specified client user ID terminated because a profile exception threshold was exceeded.
QLAUNTHDA	BIGINT	The number of times that a thread that was used by a connection from an application that is run by the specified client user ID abended.
QLAUNTHDC	BIGINT	The number of times that a thread that was used by a connection from an application that is run by the specified client user ID was canceled.
QLAUNTHD	INTEGER	The current number of active threads for the application that is run by the specified client user ID.
QLAUHTHD	INTEGER	For a statistics trace, the highest number of active threads during the current statistics interval. For a <b>READS</b> request, the highest number of active threads since DDF was started.
QLAUTHDTM	BIGINT	The number of threads that are associated with the specified client user ID that was queued because the <b>MAXDBAT</b> subsystem parameter value was exceeded.
QLAUTHDTI	BIGINT	The number of threads that are associated with the specified client user ID that was terminated because the <b>IDTHTOIN</b> subsystem parameter value was exceeded.
QLAUTHDTC	BIGINT	The number of threads that are associated with the specified client user ID that was terminated because the <b>CANCEL THREAD</b> command was issued.
QLAUTHDTR	BIGINT	The number of threads that are associated with the specified client user ID that was terminated because a profile exception condition for idle threads was exceeded.

Column name	Data type	Description
QLAUTHDTK	BIGINT	The number of threads that are associated with the specified client user ID that were terminated because the threads were running under <b>KEEPDYNAMIC</b> refresh rules, and the idle time exceeded the <b>KEEPDYNAMICREFRESH</b> idle time limit (20 minutes).
QLAUTHDTF	BIGINT	The number of threads that are associated with the specified client user ID that were terminated because the threads were running under <b>KEEPDYNAMIC</b> refresh rules, and the time that they were in use exceeded the <b>KEEPDYNAMICREFRESH</b> in-use time limit.
QLAUTHDTN	BIGINT	The number of threads that are associated with the specified client user ID that were terminated due to network termination.

The information in IFCID 412 can help identify users who are using resources inefficiently in the following types of situations:

- ▶ A user is not releasing a connection in a timely manner.
- ▶ A user is causing unexpected thread termination.
- ▶ A user is monitored by a profile, but thresholds must be adjusted (they are too high or too low).
- ▶ A user is monopolizing resources (such as opening multiple threads) and should be monitored.

If the trace identifies more than 50 client user IDs to be reported, Db2 generates multiple IFCID 412 records. For IFI READS requests, only those IFCID 412 records that fit into the buffer are returned. The interval at which the IFCID 412 statistics trace records are produced is controlled by the **STATIME\_MAIN** subsystem parameter.

These counters are maintained only for server sites and only at the client user ID level. When IFCID 412 is written as part of a statistics trace, the HWM is the maximum value that is observed since the last statistics trace interval. When IFCID 412 is created for a **READS** request, the HWM is the maximum value that was observed since DDF was started.

IFCID 412 information also is available in Db2 12 for z/OS after applying the PTF for PH40244.

### 12.2.3 New fields in IFCID 365 for recording DDF location statistics

IFCID 365 records provide detailed location statistics about the remote locations that a Db2 subsystem communicates with by using the DRDA protocol. IFCID 365 can be started by using statistics trace class 7.

Example 12-1 shows the new fields that were added to IFCID 365 since Db2 12 became generally available. They provide information about DBATs from remote locations.

*Example 12-1 New DDF location statistics fields in IFCID 365*

---

```

* FIELDS QLSTNREST TO QLSTNTHDC INCLUSIVE ARE COUNTERS WHICH WILL
* ONLY BE MAINTAINED FOR LOCATIONS WHERE Db2 IS THE SERVER AND ONLY
* FOR LOCATION STATISTICS AND DRDA REMOTE LOCS STATISTICS.
* FIELDS QLSTNREST TO QLSTFSEC CONTAIN COUNTS OF THE NUMBER OF
* OCCURRENCES OF A DETECTED CONNECTION/APPLICATION CONDITION ON
* CONNECTIONS RECEIVED FROM THE LOCATION AS FOLLOWS:
QLSTNREST DS  D      REST SERVICE REQUESTS
QLSTNSSR  DS  D      SETS OF SPECIAL REGISTERS VIA PROFILE

```

```

QLSTNSGV DS D SETS OF GLOBAL VARIABLES VIA PROFILE
QLSTNWL B DS D SYSPLEX WORKLOAD BALANCING USED
QLSTNTLS DS D TLS/SSL USED
QLSTNTRS DS D TRUSTED CONTEXT USED
QLSTNAES DS D AES ENCRYPTION USED
QLSTNXA DS D XA GLOBAL TRANSACTION USED
QLSTNENC DS D DRDA ENCRYPTION USED
QLSTNPWD DS D UID/PWD USED
QLSTNKER DS D KERBEROS USED
QLSTNMFA DS D MFA USED
QLSTNCCA DS D CLIENT CERTIFICATES USED
QLSTFSEC DS D FAILED SECURITY AUTHENTICATIONS
* FIELDS QLSTNCRSR TO QLSTSPCMT CONTAIN COUNTS OF THE NUMBER
* OF OCCURRENCES OF DETECTED CLIENT APPLICATION PROCESSING
* CONDITIONS ON CONNECTIONS RECEIVED FROM THE LOCATION
* WHICH PREVENTED Db2 FROM POOLING THE DBAT AS FOLLOWS:
QLSTHCRSR DS D WITH HOLD CURSOR NOT CLOSED
QLSTDGTT DS D DGTT NOT DROPPED
QLSTKPDYN DS D KEEP DYNAMIC PACKAGES USED
QLSTHIPRF DS D HIGH-PERFORMANCE DBATS USED *
QLSTHLOBLOC DS D HELD LOB LOCATOR(S) EXIST
QLSTSPCMT DS D STORED PROC COMMIT PERFORMED
* FIELDS QLSTCNVTC TO QLSTCNVTQW CONTAIN COUNTS OF THE NUMBER
* OF OCCURRENCES WHERE CONNECTION REQUESTS RECEIVED FROM THE
* LOCATION WERE TERMINATED BY Db2 DUE TO THE FOLLOWING:
QLSTCNVTC DS D CONDBAT REACHED
QLSTCNVTP DS D PROFILE EXCEPTION
QLSTCNVTQN DS D MAXCONQN REACHED
QLSTCNVTQW DS D MAXCONQW REACHED
* FIELDS QLSTNTHDPQ TO QLSTNTHDC CONTAIN COUNTS OF THE NUMBER
* OF OCCURRENCES WHERE THREADS USED BY CONNECTIONS RECEIVED
* FROM THE LOCATION WERE QUEUED/TERMINATED AS FOLLOWS:
QLSTNTHDPQ DS D QUEUED - PROFILE EXCEPTION
QLSTNTHDPT DS D TERMINATED - PROFILE EXCEPTION
QLSTNTHDA DS D ABENDED
QLSTNTHDC DS D CANCELED
* QLSTNCNV TO QLSTHTHD INCLUSIVE ARE MEANINGFUL AT SERVER ONLY
* AND REPRESENT AT THE TIME OF A GENERATED STATISTIC TRACE RECORD
* OR A READS REQUEST THE CURRENT AND HIGH WATER MARK OF CONNECTIONS
* AND THREADS ATTRIBUTED TO THE LOCATION. FOR A STATISTICS TRACE,
* THE HIGH WATER MARK IS THE MAXIMUM OBSERVED VALUE SINCE THE LAST
* STATISTICS TRACE INTERVAL REQUEST. FOR A READS REQUEST, THE HIGH
* WATER MARK IS THE MAXIMUM OBSERVED VALUE SINCE DDF WAS STARTED.
QLSTCONV DS OD CONNECTIONS FROM LOCATION
QLSTNCNV DS F CURRENT NUMBER OF CONNECTIONS
QLSTHCNV DS F CONNECTIONS HIGH WATER MARK *
QLSTTHRD DS OD ACTIVE THREADS (DBATS) FOR LOCATION
QLSTNTHD DS F CURRENT NUM OF ACTIVE THREADS
QLSTHTHD DS F ACTIVE THREADS HIGH WATER MARK *
QLSTNTPLH DS D TERMINATED - HIGH PERFORMANCE
*
* DBAT REMAINED IN POOL LONGER
* THAN POOLINAC (OR 120 SECONDS
* IF POOLINAC=0)

```

QLSTNTILS DS D TERMINATED - TCP SOCKET CLOSED  
 \* DUE TO CONNECTION LOSS

**Note:** QLSTNTPLH and QLSTNTILS are available in Db2 13 only, but the other fields also are available in Db2 12.

As with IFCID 411 and 412, if the trace identifies more than 50 locations to be reported, Db2 generates multiple IFCID 365 records. For IFI READS requests, only those IFCID 365 records that fit into the buffer are returned. The interval at which the IFCID 365 statistics trace records are produced is controlled by the **STATIME** subsystem parameter.

Because these new fields are part of the QLST data section, and the DSNDQLST section also is included in IFCID 1, these extra fields also are available in the standard Db2 IFCID 1 statistics record.

### 12.2.4 New fields in IFCID 1 for recording global DDF activity

New fields were added to IFCID 1 in Db2 13 to provide information about DBATs that remained active or that were terminated at the server. This new information can help you analyze resource usage at the server.

Table 12-3 shows these new IFCID 1 fields.

Table 12-3 New global DDF statistics fields in IFCID 1

Column name	Data type	Description
QDSTNAKD	INTEGER	The current number of DBATs that are active due to the usage of packages that are bound with <b>KEEPDYNAMIC(YES)</b> .
QDSTMAKD	INTEGER	The maximum number of DBATs that are active due to the usage of packages that were bound with <b>KEEPDYNAMIC(YES)</b> since DDF was started.
QDSTNDBT	INTEGER	The number of DBATs that terminated since DDF was started.
QDSTNTPL	INTEGER	The number of DBATs that terminated after remaining in pool longer than <b>POOLINAC</b> since DDF was started.
QDSTNTRU	INTEGER	The number of DBATs that terminated after being reused more times than the limit since DDF was started.
QDSTDBPQ	INTEGER	The current number of DBATs that were suspended due to a profile exception.
QDSTMDPQ	INTEGER	The maximum number of DBATs that were suspended due to a profile exception since DDF was started.

**Note:** The changes that include the two new fields QDSTDBPQ and QDSTMDPQ, which were introduced by APAR [PH47626](#), become available when FL V13R1M500 or later is activated in Db2 13. Also, the existing QDSTNDBA field was updated in Db2 13. QDSTNDBA represents the number of times that a DBAT was created. This value did not include DBATs that were created to replace disconnected (pooled) DBATs that terminated because they reached their reuse limit. The DBAT creation statistics counter now always counts the number of DBATs that are created.

## 12.2.5 Db2 AI for z/OS 1.5 enhancements for DDF

Db2 AI for z/OS 1.5 (Db2ZAI 1.5) uses the extra statistics that are provided by IFCID 365, IFCID 411, and IFCID 412 as input to the training process, the results of which are used to generate warning and exception profiles for connections or threads.

In Db2 AI for z/OS 1.4 (Db2ZAI 1.4), the Distributed Connection Control (DCC) component recommends profiles (**MONITOR THREADS/CONNECTIONS**) based on IP address (requester) by using the information that is collected in IFCID 365. With the more granular client information that is provided by IFCID 411 or 412, Db2ZAI 1.5 DCC can recommend profiles (**MONITOR THREADS**) based on extra filtering categories (**CLIENT\_APPLNAME** and **CLIENT\_USERID**), as shown in Figure 12-1, Figure 12-2, and Figure 12-3 on page 352.

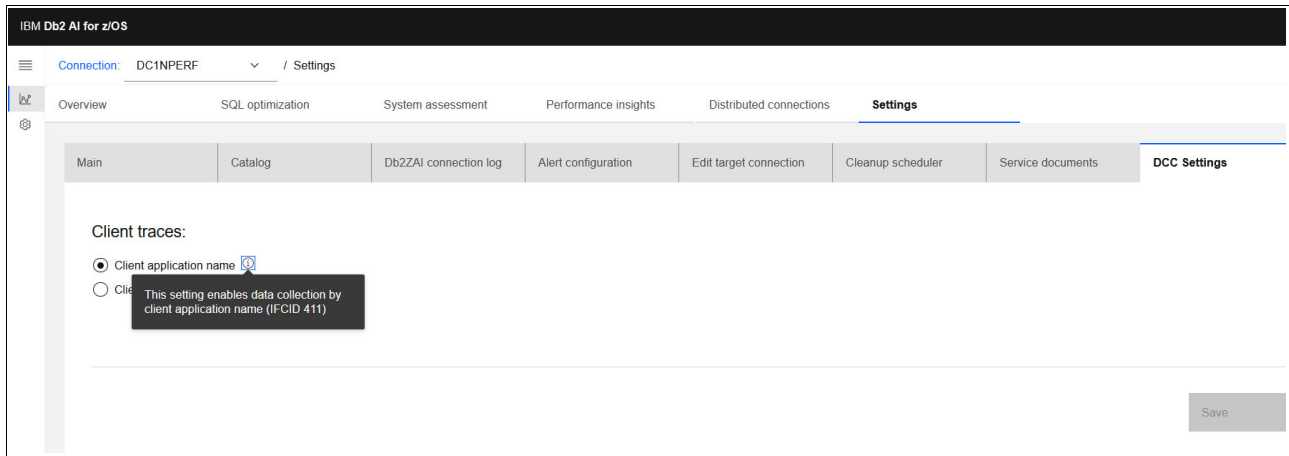


Figure 12-1 Db2ZAI new DCC settings for client application name

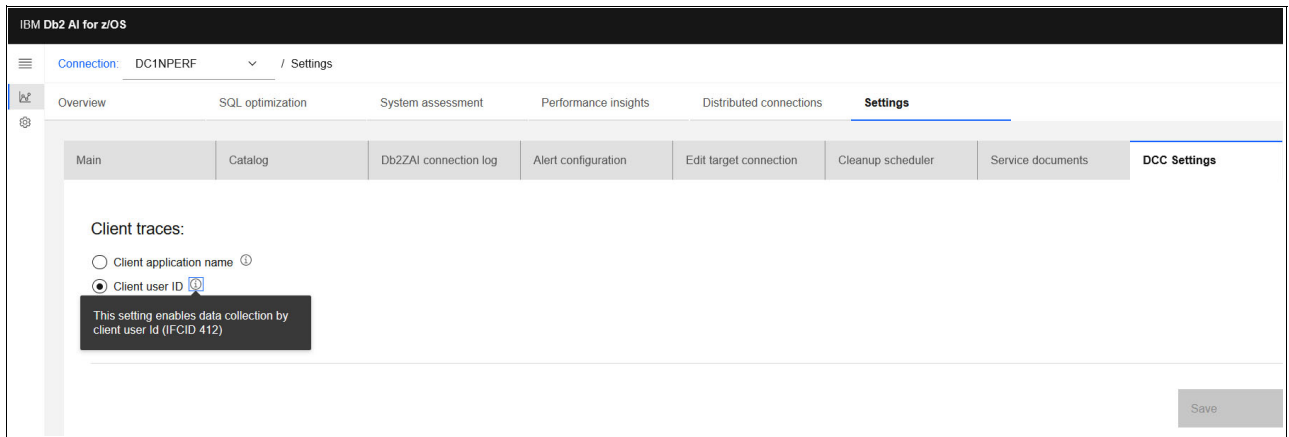


Figure 12-2 Db2ZAI new DCC settings for client user ID

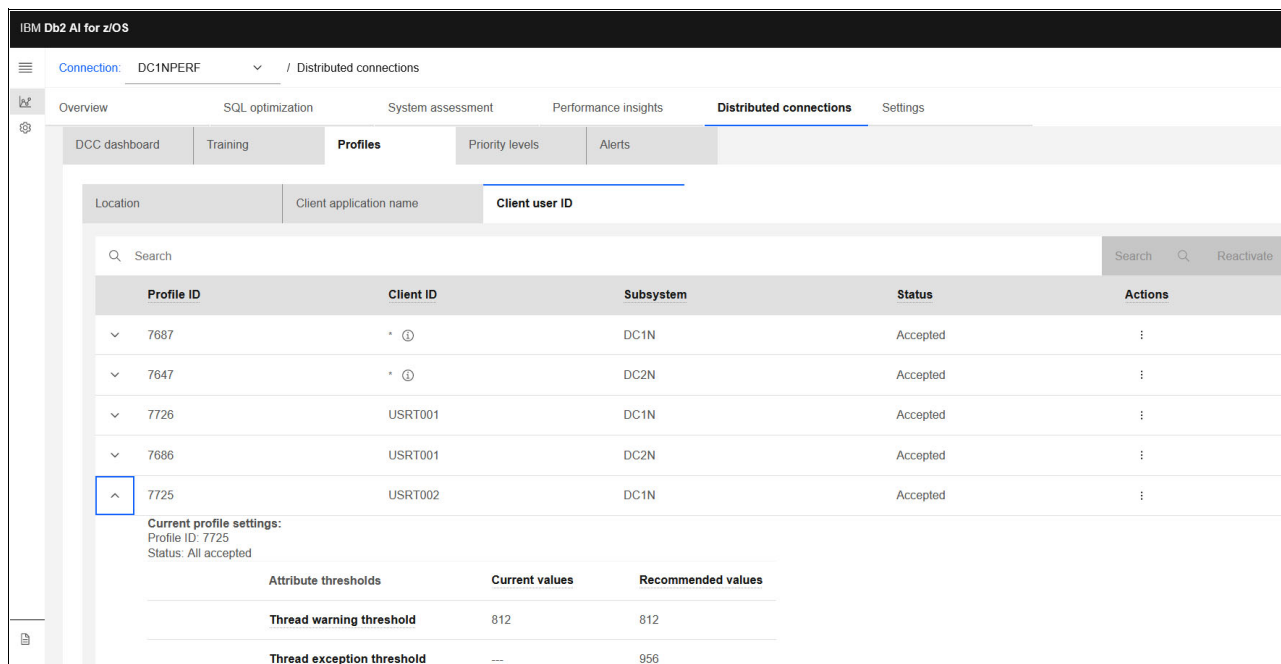


Figure 12-3 Db2ZAI new profile filtering categories

## 12.3 IFCID 003: Recording the longest wait time for certain suspension types

To maintain data consistency and concurrency control, Db2 uses locks and latches to serialize access to resources. However, there are cases when the resources are held for too long, which can cause performance issues that must be addressed. Before Db2 13, to find these long latch and lock suspensions, extra and often high-volume traces are required. These extra traces sometimes result in thousands of records, which make it difficult to analyze the information.

To narrow down the searches, it is important to know when the longest lock or latch suspensions occurred and for how long. Therefore, Db2 13 added an IFCID 3 section that includes the longest suspension time for a select number of suspensions that occurred during the life of the thread. Besides information about the longest lock or latch wait, Db2 also records the longest service task wait information, and the longest page latch wait information. The longest wait time for each of these suspensions recorded, and information about the resource that experienced the contention also is reported.

A new section, QWA01REO, was added to IFCID 3 and is mapped by DSNDQLLL to provide this extra information. This control block contains information about the longest internal resource lock manager (IRLM) lock, Db2 latch, log, or database I/O wait; the longest service task wait; and the longest page latch wait.

APAR [PH46371](#) adds an extra field that is called QLLLTY, which makes it easier to identify the type of suspension that is reported on in the first part of the QLLL section. APAR [PH46372](#) adds support for roll-up records to QLLL. When ACCUMAC >= 2, the roll-up records record the largest wait times for parts 1, 2, and 3 independently. This information is recorded each time that a roll-up accumulation occurs.



Example 12-2 shows the fields of this new QWA01REO section that is mapped by DSNDQLL.

Example 12-2 Updated IFCID 3 with new section QWA01REO

---

QLLLH	DS	0CL8	QLLL HEADER SECTION
QLLLHID	DS	CL2	QLLL BLOCK ID
QLLLHLEN	DS	XL2	QLLL BLOCK LENGTH
QLLLHEYE	DS	CL4	QLLL BLOCK EYE CATCHER
***** PART 1 LONGEST L/L,S/A,DL INFORMATION *****			
QLLLR	DS	0CL36	LONGEST L/L,S/A,DL INFORMATION
QLLLLH	DS	0XL4	LOCK HASH QW0044LH
QLLLLDB	DS	XL2	DBID FOR I/O
QLLLLOB	DS	0XL2	OBID FOR I/O
QLLLLC	DS	XL2	LATCH CLASS
QLLLLK	DS	0CL32	LOCK NAME QW0044LK
QLLLLA	DS	XL8	LATCH TOKEN
	DS	CL24	RESERVED
QLLLFLAG	DS	0CL4	QLLL FLAGS
QLLLTYP	DS	XL1	WAIT TYPE FOR PART 1
QLLLAWLG	EQU	X'01'	LOG WRITE I/O
QLLLAWTI	EQU	X'02'	Db2 - I/O
QLLLAWTD	EQU	X'03'	SYNCH DB READS WITH DASD
*			CACHE HIT
QLLLAWTG	EQU	X'04'	SENDING OF IRLM NOTIFY MSG
QLLLAWTN	EQU	X'05'	PAGESET/PARTITION P-LOCKS
QLLLAWTO	EQU	X'06'	PAGE P-LOCKS
QLLLAWTQ	EQU	X'07'	OTHER P-LOCKS
QLLLAWTJ	EQU	X'08'	PARENT L-LOCKS
QLLLAWTK	EQU	X'09'	CHILD L-LOCKS
QLLLAWTM	EQU	X'0A'	OTHER L-LOCKS
QLLLAWTL	EQU	X'0B'	DB2 - LOCK
QLLLAWLH	EQU	X'0C'	DB2 - LATCH
QLLLAWTR	EQU	X'0D'	DB2-READ I/O UNDER THREAD
*			OTHER THAN THIS ONE
QLLLAWTW	EQU	X'0E'	DB2-WRITE I/O UNDER THREAD
*			OTHER THAN THIS ONE
QLLLAWDR	EQU	X'0F'	DRAIN LOCK
	DS	CL3	RESERVED
QLLLRT	DS	0CL24	BEGIN/END/DELTA TIME
QLLLRBT	DS	XL8	PART 1 QLLLTYP BEGIN TIME
QLLLRET	DS	XL8	PART 1 QLLLTYP END TIME
QLLLRDT	DS	XL8	PART 1 QLLLTYP DELTA TIME
***** PART 2 LONGEST SERVICE TASK WAIT *****			
QLLLS	DS	0CL8	LONGEST SERVICE TASK WAIT
QLLLID	DS	C	RMID
QLLLFC	DS	XL2	FUNCTION CODE
	DS	CL5	RESERVED
QLLLST	DS	0CL24	
QLLLSBT	DS	XL8	SERVICE TASK BEGIN TIME
QLLLSET	DS	XL8	SERVICE TASK END TIME
QLLLSDT	DS	XL8	SERVICE TASK DELTA TIME
***** PART 3 LONGEST PAGE LATCH INFORMATION *****			
QLLLPI	DS	0CL16	PAGE LATCH WAIT INFORMATION
QLLLNTID	DS	0CL4	PAGESET IDENTIFIER
QLLLPIDB	DS	XL2	DATABASE ID

QLLLPIB	DS	XL2	PAGESET ID
QLLLPG	DS	XL4	PAGE NUMBER
QLLLPA	DS	XL4	PART NUMBER
	DS	CL4	RESERVED
QLLLPI	DS	OCL24	BEGIN/END/DELTA TIME
QLLLPIBT	DS	XL8	PAGE LATCH BEGIN TIME
QLLLPIET	DS	XL8	PAGE LATCH END TIME
QLLLPIDT	DS	XL8	PAGE LATCH DELTA TIME
QLLLRACE	DS	XL8	ACE for part 1 of QLLLTYP Wait
QLLLSACE	DS	XL8	ACE for part 2 of Service Task Wait
QLLLPIACE	DS	XL8	ACE for part 3 of Page Latch Wait
	DS	OCL8	
QLLLRUID	DS	CL2	ROLLUP ID
*			/*RU FOR RRSF AND DDF ROLLUP */
*			/*PQ FOR PARALLEL QUERIES */
*			/*AT FOR AUTONOMOUS TRANSACTION*/
*	DS	CL38	RESERVED
*			
QLLLEND	DS	OC	
		MEND	

---

Figure 12-4 shows the layout from the IBM OMEGAMON for Db2 Performance Expert (OMPE) RECTRACE report of this new section QWA01REO.

LOCK/LATCH, SYNC/ASYNC I/O, DRAIN LOCK:		SERVICE TASK:	
-----		-----	
DBID FOR SYNC/ASYNC I/O.: 681		RMID.....:	10
OBID FOR SYNC/ASYNC I/O.: 7		FUNCTION CODE.....:	89
LOCK HASH.....:	X'02A90007'	ACE ADDRESS.....:	N/P
LATCH CLASS.....:	X'0007'	BEGIN TIME.....:	10/24/22 04:12:14.992017
LATCH TOKEN.....:	N/P	END TIME.....:	10/24/22 04:12:15.071715
ACE ADDRESS.....:	N/P	DELTA TIME.....:	0.079698
WAIT TYPE.....:	SYNCH DB READS WITH DASD HIT		
BEGIN TIME.....:	10/24/22 04:12:13.844473		
END TIME.....:	10/24/22 04:12:13.853016		
DELTA TIME.....:	0.008543		
PAGE LATCH WAIT INFORMATION:		OTHER INFORMATION:	
-----		-----	
DATABASE ID.....:	DSNDB06	ROLLUP ID.....:	N/P
PAGESET ID.....:	DSNDXX07		
ACE ADDRESS.....:	N/P		
PAGE NUMBER.....:	57		
PARTITION NUMBER.....:	N/P		
BEGIN TIME.....:	10/24/22 04:12:14.403778		
END TIME.....:	10/24/22 04:12:14.403948		
DELTA TIME.....:	0.000170		
LOCK/LATCH, SYNC/ASYNC I/O, DRAIN LOCK:		SERVICE TASK:	
-----		-----	
DBID FOR SYNC/ASYNC I/O.: DSNDB06		RMID.....:	3
OBID FOR SYNC/ASYNC I/O.: DSNDDXX01		FUNCTION CODE.....:	73
LOCK HASH.....:	X'0006005F'	ACE ADDRESS.....:	N/P
LATCH CLASS.....:	X'005F'	BEGIN TIME.....:	10/24/22 06:48:38.254521
LATCH TOKEN.....:	X'0000005D87EFB500'	END TIME.....:	10/24/22 06:48:38.254603
ACE ADDRESS.....:	N/P	DELTA TIME.....:	0.000082
WAIT TYPE.....:	DB2-READ I/O UNDER THRD		
BEGIN TIME.....:	10/24/22 06:48:38.074237		
END TIME.....:	10/24/22 06:48:38.075099		
DELTA TIME.....:	0.000862		
PAGE LATCH WAIT INFORMATION:		OTHER INFORMATION:	
-----		-----	
DATABASE ID.....:	0	ROLLUP ID.....:	N/P
PAGESET ID.....:	0		
ACE ADDRESS.....:	N/P		
PAGE NUMBER.....:	N/P		
PARTITION NUMBER.....:	N/P		
BEGIN TIME.....:	N/P		
END TIME.....:	N/P		
DELTA TIME.....:	N/P		

Figure 12-4 Sample layout of IFCID 3: Longest lock or latch waiter

## 12.4 Group buffer pool residency time enhancements

IBM z16 introduces the ability to track data and directory residency statistics for group buffer pools (GBPs).

Two new metrics were added to relevant GBP statistics storage areas and are accessible through extra fields in IFCID 230 and IFCID 254 trace records, as shown in Figure 12-4 on page 355 and Table 12-5.

Table 12-4 New group buffer pool residency statistics fields in IFCID 230

Column name	Data type	Description
QBGBART	CHAR(8)	Data Area Residency Time: The weighted average, in microseconds, of the elapsed time that a data area is in a GBP before the data area is reclaimed.
QBGBERT	CHAR(8)	Directory Entry Residency Time: The weighted average, in microseconds, of the elapsed time that a directory entry is in a GBP before the directory entry is reclaimed.

Table 12-5 New GBP residency statistics fields in IFCID 254

Column name	Data type	Description
QW0254AR	CHAR(8)	Data Area Residency Time: The weighted average, in microseconds, of the elapsed time that a data area is in a GBP before the directory entry is reclaimed.
QW0254ER	CHAR(8)	Directory Entry Residency Time: The weighted average, in microseconds, of the elapsed time that a directory entry is in a GBP before the data area is reclaimed.

You also can display the same information by using the **-DISPLAY GROUPBUFFERPOOL** command.

When the **-DISPLAY GROUPBUFFERPOOL** command includes the **GDETAIL** option, the output includes statistics on the residency times for items in the GBP with a new DSNB820I message. Example 12-3 shows a sample DSNB820I message.

Example 12-3 Sample DSNB820I message

---

```
DSNB820I  -DBCL  AVERAGE RESIDENCY TIME IN MICROSECONDS 756
              FOR DIRECTORY ENTRIES                      = 19984895
              FOR DATA ENTRIES                          = 5446695
```

---

For more information about the measurements that were conducted to understand the performance effects of GBP residency time in a data-sharing environment, see Chapter 3, “Synergy with the IBM Z platform” on page 43.

## 12.5 IFCID 369 enabled by default when STATISTICS CLASS(1) starts

IFCID 369 (aggregated accounting statistics) was introduced in Db2 10 to aggregate accounting statistics per connection type. Until now, collecting IFCID 369 statistics was optional. They could be collected by explicitly issuing the **-START TRACE(STAT)CLASS(9)** command.

In Db2 13, IFCID 369 was added to statistics trace class 1 (-START TRACE(STAT) CLASS(1)). Because most customers start statistics trace class 1 at Db2 startup by using the SMFSTAT subsystem parameter by default, IFCID 369 statistics now always are collected. As before, the collection of IFCID 369 statistics requires that Db2 accounting traces also are started. To maintain compatibility, -START TRACE(STAT)CLASS(9) remains unchanged and can still be used to collect IFCID 369 statistics. (When both trace classes are active, Db2 collects and writes the information only once.)

This enhancement is also available in Db2 12 for z/OS after applying the PTF for PH43916.

## 12.6 IFCID changes for application timeout and deadlock control

A number of IFCID changes were made to support the Db2 13 enhancements to enable application-level lock timeout and deadlock control.

Two new counters were added to IFCID 2 (statistics trace record) and IFCID 3 (accounting trace record), as shown in Example 12-4.

*Example 12-4 New lock timeout fields in IFCID 2 and IFCID 3*

---

QXSTTIMEOUTFromApp1	DS D	# times a SET CURRENT LOCK TIMEOUT SQL statement was executed
QXSTTIMEOUTFromProf	DS D	# times a SET CURRENT LOCK TIMEOUT SQL was executed from the DSN_PROFILE_TABLE

---

IFCID 106 (subsystem parameters trace record) was modified to include a new field, QWP4LTMX, to capture the setting of the new SPREG\_LOCK\_TIMEOUT\_MAX subsystem parameter, as shown in Example 12-5.

*Example 12-5 New lock timeout field in IFCID 106*

---

QWP4LTMX	DS FL2	SPREG_LOCK_TIMEOUT_MAX (-1, 0-32767)
----------	--------	--------------------------------------

---

A new flag, QW0172WAS, was added to IFCID 172 (deadlock trace record) to record whether the “worth value” was set by using the global variable or another method. The new field is shown in Example 12-6.

*Example 12-6 New flag in IFCID 172*

---

QW0172WAS	DS CL1	SOURCE OF THE WORTH VALUE
		G-GLOBAL VARIABLE
		O-OTHER

---

IFCID 196 (timeout trace record) was updated to record the timeout interval and the source of the timeout interval setting, as shown in Example 12-7.

*Example 12-7 New timeout interval fields in IFCID 196*

---

QW0196TI	DS F	TIMEOUT INTERVAL
QW0196TR	DS CL1	Source of QW0196TI interval
		Z -IRLMRWT
		S -Special register
		I -IRLM internal
QW0196HTI	DS F	Holdings TIMEOUT interval

---

QW0196HTR DS CL1 Source of QW0196HT interval  
 Z -IRLMRWT  
 S -Special register  
 I -IRLM internal

---

IFCID 437 is a new trace record that records the usage of the **SET CURRENT LOCK TIMEOUT** statement, and whether it is set directly by the application or by using a system profile. Example 12-8 shows the layout of the new trace record.

*Example 12-8 New IFCID 437 trace record*

---

QW0437	DSECT		
QW0437FromSource	DS	CL1	SET was done by application or Profile table
QW0437FromApp1	EQU	C'A'	SET was done by application
QW0437FromProfile	EQU	C'P'	SET was done by Profile table
QW0437OldTimeOutNI	DS	CL1	Null indicator for old value
	DS	CL2	Available
QW0437OldTimeOut	DS	F	Old value if not null
QW0437NewTimeOutNI	DS	CL1	Null indicator for new value
	DS	CL3	Available
QW0437NewTimeout	DS	F	New value if not null
QW0437Status	DS	CL1	Status
QW0437CS	EQU	C'S'	SUCCESSFUL STMT – Constant
* for QW0437Status successful status on statement execution			
QW0437CF	EQU	C'F'	FAILED STMT – Constant
* for QW0437Status Failed status on statement execution			
	DS	CL3	Available
*			
* The following fields are applicable only			
* when QW0437FromSource = QW0437FromApp1			
*			
QW0437Co11ID_Off	DS	H	Offset to collection ID
QW0437PKG_Off	DS	H	Offset to package ID
QW0437Contoken	DS	CL8	Package consistency token
*			
QW0437Co11IDDSECT			
QW0437Co11ID_len	DS	H	Collection ID length
QW0437Co11ID_var	DS	CL128	%U Collection ID
*			
QW0437PKG DSECT			
QW0437PKG_len	DS	H	Package ID length
QW0437PKG_var	DS	CL128	%U Package ID

## 12.7 Partition range support for IFCID 306 READS requests

High log reader throughput is essential for replication operations, especially when you must optimize your continuous availability solutions for minimal recovery point objectives (RPOs) and recovery time objectives (RTOs). Currently, retrieving logs from a range of partitions can take a long time for IFCID 306 to process your request.

Db2 13 adds more flexibility when using **READS** for IFCID 306 so that you can use partition filtering for a range of partitions to speed up the log read process and increase the throughput for replication operations. This enhancement benefits replication products that run multiple capture programs in which each capture instance processes a different set of partitions of the same table space in parallel for better end-to-end performance.

A new **WQLSPTRG** flag is defined in the qualification area fields of WQLS. When you turn on this flag, the mapping of WQLSDBPS in WQLS is replaced with the new mapping of WQLSDBPP. The WQLSDBPP mapping contains two new fields that are called WQLSPPART\_LOW and WQLSPPART\_HIGH, which can be specified only once for each table space.

Table 12-6 lists and describes the qualification area fields of WQLSDBPP. This area is mapped by the assembly language mapping macro DSNDWQAL. This DSECT maps the area that is used to filter IFCID 306 records when WQLSTYPE is set to 'DBPP'.

*Table 12-6 WQLSDBPP fields*

<b>Name</b>	<b>Hex offset</b>	<b>Datatype</b>	<b>Value description</b>
WQLSPDBID	0	Hex, 2 bytes	DBID of the database
WQLSPPSID	2	Hex, 2 bytes	Pageset ID (PSID) of the partitioned table space
WQLSPP ART_LOW	4	Hex, 2 bytes	Low partition number of the table space partition range
WQLSPP ART_HIGH	6	Hex, 2 bytes	High partition number of the table space partition range

For a partitioned table space, you can set WQLSPPART\_LOW and WQLSPPART\_HIGH to any value of low and high available partition numbers. If WQLSPPART\_LOW is '0000'x and WQLSPPART\_HIGH is '0000'x or 'FFFF'x, the entire partitioned table space qualifies for the IFCID 306 process.

For a non-partitioned table space, you must set both WQLSPPART\_LOW and WQLSPPART\_HIGH to zero.

## 12.8 New fields in IFCID 3 for SQL Data Insights

Four new fields were added to the IFCID 3 accounting trace record in the DSNDQWAC section in support of SQL Data Insights, as shown in Example 12-9. They provide information about the following items:

- ▶ The elapsed time that was spent performing SQL Data Insights functions
- ▶ The general-purpose CPU time that was spent in SQL Data Insights functions
- ▶ The IBM zSystems Integrated Information Processor (zIIP) time that was spent in SQL Data Insights functions
- ▶ The number of invocations of SQL Data Insights functions

The elapsed and CPU times are included in regular class 2 elapsed and CPU times and represent only those portions that are spent running the built-in scalar SQL Data Insights functions.

*Example 12-9 New SQL Data Insights related IFCID 3 fields*

---

```

QWAC_AIDB_FNS_ELAP DS CL8 /* The accumulated elapsed time spent */
*                          /* processing SQL Data Insight functions */
*                          /* This time is included in Class 2 */
*                          /* elapsed time. */
QWAC_AIDB_FNS_CP DS CL8 /* The accumulated CPU time spent */
*                          /* processing SQL Data Insights functions */
*                          /* This time is included in Class 2 */
*                          /* CPU time. It does not include CPU */
*                          /* consumed on an IBM specialty engine. */
QWAC_AIDB_FNS_zIIP DS CL8 /* The accumulated CPU time spent */
*                          /* processing SQL Data Insights functions */
*                          /* on an IBM specialty engine. This time */
*                          /* is included in Class 2 CPU time. */
QWAC_AIDB_COUNT DS XL8 /*The number of entry/exit events */
*                          /* performed by SQL Data Insights functions*/

```

---

For more information about SQL Data Insights, see Chapter 6, “SQL Data Insights” on page 211.

## 12.9 New fields in IFCID 2 for plan authorization cache

Two new counters were added to IFCID 2 to improve the monitoring of plan authorization caching, as shown in Example 12-10. Before Db2 13, this type of information was available only for packages and routines.

*Example 12-10 New plan authorization cache-related IFCID 2 fields*

---

```

QTAUCNOT DS F #PLAN AUTH CHECKS THAT COULD NOT *
MAKE USE OF THE PLAN AUTH CACHE
QTAUCOW1 DS F # OF TIMES DB2 OVERWROTE AN AUTHID *
IN THE PLAN AUTH CACHE

```

---



## 12.10 New field in IFCID 389 for fast index traversal

Trace record IFCID 389 is used to record all indexes that use fast index traversal (fast traversal blocks (FTB)). A new field, FTB Factor, was added to IFCID 389, as shown in Example 12-11. It is calculated as the modified moving average count of the number of traversals. This value is decreased for every sequential access and for every index page split.

*Example 12-11 New FTB Factor field in IFCID 389*

---

```
QW0389FF DS    XL4 FTB FACTOR
```

---

You also can display the FTB Factor value by using the **-DISPLAY STATS(INDEXTRAVERSECOUNT)** command, as shown in the DSNT830I message in the Example 12-12.

*Example 12-12 Sample DSNT830I message*

---

```
-DIS STATS(ITC) DB(DB1) LIMIT(5)  
Output (traverse counts are in descending order)  
DBID PSID DBNAME IX-SPACE LVL PART TRAV. COUNT FTB FACTOR  
0017 0005 DB1    IX1      003 0001 00000050099 -0000000016  
0017 0005 DB1    IX2      003 0010 00000050001 -0000010032  
0017 0005 DB1    IX2      003 0008 00000030021 00000030021  
0017 0005 DB1    IX2      003 0016 00000029999 00000029999  
0017 0005 DB1    IX3      003 0001 00000000999 00000000999  
***** DISPLAY OF STATS ENDED *****
```

---

For more information about FTB-related performance measurements, see Chapter 2, “Scalability and availability” on page 13.

## 12.11 IFCID changes that are related to storage manager large object contraction

To improve the monitoring of how Db2 manages above-the-bar (ATB) storage consumption, several IFCIDs were updated in Db2 13.

The fields that are shown Example 12-13 are no longer used and were removed from the QSST section of IFCID 1.

*Example 12-13 Removed QSST IFCID 1 fields*

---

```
0001 QSST_P64DISNUM      (S) NUMBER OF 64-BIT POOLS THAT WERE CONTRACTED.  
0001 QSST_P64DISBLK     (S) NUMBER OF 64-BIT POOL BLOCKS THAT REQUIRED A DISCARD.
```

---

The three new fields that are shown in Example 12-14 were added to the QSST section of IFCID 1.

*Example 12-14 New storage manager large object contraction-related IFCID 1 fields*

---

0001 QSST_P64PCTRCT	(S) NUMBER OF 64-BIT POOLS THAT WERE CONTRACTED.
0001 QSST_P64DISNO	(S) NUMBER OF TIMES THAT IARV64 REQUEST=DISCARDATA,KEEPREAL(NO) WAS ISSUED.
0001 QSST_P64DISYES	(S) NUMBER OF TIMES THAT IARV64 REQUEST=DISCARDATA,KEEPREAL(YES) WAS ISSUED.

---

These fields track the number of contractions of ATB storage pools and the number of actual **DISCARDATA** requests that are performed, either with **KEEPREAL(NO)** or **KEEPREAL(YES)**.

For more information about the performance tests that were conducted to measure the effect of the ATB storage contraction optimization in Db2 13, see Chapter 2, “Scalability and availability” on page 13.

## 12.12 IFCID changes that are related to latch-level expansion

Before Db2 13, each Db2 resource that is serialized with a latch is assigned a latch type. These latch types are arranged in a hierarchy with 32 levels. Each latch type maps to a level number 1 - 32 that is called the *latch class*. Up to six latch types can map to the same latch class.

Db2 added many new latch types over the years. Several latch classes reached or are close to reaching the maximum number of latch types, which can cause latch hierarchy violations and result in errors and abends.

Db2 13 increased the number of latch classes from 32 to 64 to help avoid latch hierarchy violations and provide better diagnostics.

The following IFCIDs were changed to support this latch classes expansion:

- ▶ In IFCID 1, the DSNDQVLS sections were expanded to include the new latch classes from 33 to 64.
- ▶ In IFCID 3, in the DSNDQLLL section, the latch class field QLLLLC was extended from FIXED(08) to FIXED(16).
- ▶ IFCID 51 extended the latch class in the Q0051LC field from FIXED(8) to FIXED(16).
- ▶ IFCID 52 extended the latch class in the Q0052LC field from FIXED(8) to FIXED(16).
- ▶ IFCID 56 extended the latch class in the Q0056LC field from FIXED(8) to FIXED(16).
- ▶ IFCID 57 extended the latch class in the Q0057LC field from FIXED(8) to FIXED(16).
- ▶ IFCID 148 extended the latch class in the Q0148LC field from FIXED(8) to FIXED(16).

## 12.13 Obtaining the most current information about IFCID changes

This chapter documents the changes and enhancements that were known at the time of writing. Db2 for z/OS now follows a continuous delivery model, and further changes and enhancements in the instrumentation facility area are likely to be introduced throughout the life of Db2 13. For a continually updated summary of all IFCID changes in Db2 13, see [IFCID changes](#) or the same section in *Db2 13 for z/OS: What's New?*

The latest IFCID trace field mappings file (DSNWMSGs) now also is available online with more frequent updates. As in earlier Db2 releases, the DSNWMSGs file also remains available in a data set that is included with Db2 product. However, any updates to the DSNWMSGs file in this location require APARs, and for various practical reasons, these APARs cannot be released with the same frequency as the related enhancements in continuous delivery. So, the DSNWMSGs file is now available in two different locations, but the online version is always more current:

- ▶ Customers who have Db2 for z/OS licenses that are associated with their IBMid can download the latest DSNWMSGs file in the PDF format from [Db2 13 for z/OS IFCID flat file \(DSNWMSGs\)](#).
- ▶ Customers can use the TSO or ISPF browse function to look at the field descriptions in the *prefix.SDSNIVPD*(DSNWMSGs) data set that is provided with Db2. However, this version is updated only infrequently by APARs that consolidate changes from continuous delivery.

## 12.14 Summary

Db2 13 introduces some new instrumentation and serviceability capabilities to support changes to your overall Db2 for z/OS environment, including support for SQL Data Insights, Db2ZAI, and the ability to monitor GBPs by leveraging changes to IBM z16. Db2 13 also expands on existing instrumentation and serviceability capabilities to keep pace with new and enhanced Db2 features in the areas of **INSERT** performance, thread monitoring, locks and latches, application timeouts and deadlocks, replication operations, plan authorization cache, fast index traversal, and large objects (LOBs).

All these changes enable you to gain insights into the behavior of your Db2 subsystems at a level that was not possible before.





# IBM Db2 AI for z/OS benefits and capacity requirement

IBM Db2 AI for z/OS (Db2ZAI) leverages the artificial intelligence (AI) capabilities of IBM Watson® Machine Learning for z/OS (WMLz) and transforms IBM Db2 for z/OS into a cognitive enterprise data management system. Through machine learning (ML), Db2ZAI can detect the existence of patterns in your unique operating environment. Based on these learned patterns, Db2ZAI provides three main use cases to help improve your Db2 for z/OS applications:

- ▶ SQL optimization that enables the Db2 for z/OS optimizer to choose better access paths for your SQL queries based on the knowledge that Db2ZAI feeds to it.
- ▶ System assessment (SA) that automatically identifies abnormal system behavior and advises Db2 system administrators on ways to tune for better Db2 system and workload performance.

Db2ZAI 1.5.0 adds a performance insights function that enhances the SA use case by providing a series of reports through a GUI to observe the state of your Db2 subsystems in real time.

- ▶ Distributed Connection Control (DCC) helps you monitor distributed Db2 connections and threads and provides a mechanism for managing them across your remote clients.

To learn more about Db2ZAI 1.5.0, see *IBM Db2 13 for z/OS and More*, SG24-8527, which provides an overall introduction to the product, or see [Db2 AI for z/OS 1.5.0](#).

This chapter includes the following topics:

- ▶ How Db2ZAI helps reduce application cost: SQL optimization
- ▶ How Db2ZAI helps with Db2 subsystem management: System assessment and performance insights
- ▶ How Db2ZAI helps manage inbound network traffic: DCC
- ▶ Summary of capacity requirements and general recommendations
- ▶ Detailed capacity evaluations for Db2ZAI 1.5
- ▶ Monitoring information

## 13.1 How Db2ZAI helps reduce application cost: SQL optimization

Db2ZAI SQL optimization can learn baseline access paths of both dynamic and static SQL statements and record the statements' baseline performance. During baseline data collection, SQL optimization records the inputs to the Db2 optimizer based on real-time runs of the SQL statements: It collects host variable and parameter marker values and remembers the screen-scrolling behavior of applications, and it also remembers data that is related to query parallelism usage.

After baseline data collection completes, Db2ZAI SQL optimization starts training with the data and feeds the learned results to the Db2 optimizer to choose the most efficient access paths for each SQL statement.

There are currently three types of models that Db2ZAI can use for SQL access paths optimization:

<b>Host variable model</b>	With the learned host variables and parameter marker values from baseline SQL runs, Db2ZAI helps the Db2 optimizer to get more accurate filter factor estimations for choosing better access paths.
<b>Screen scrolling model</b>	With the learned screen-scrolling behavior from baseline SQL runs, Db2ZAI helps the Db2 optimizer to optimize the access paths for retrieving some rows of data.
<b>Parallelism model</b>	Db2ZAI provides the Db2 optimizer with query parallelism inputs to activate proper levels of parallelism for best runtime performance.

Db2ZAI automatically can stabilize simple dynamic statements, which help applications reduce prepare overhead.

Also, Db2ZAI has a powerful feature that is built on its baseline learning about SQL statements so that Db2ZAI can detect and resolve performance regressions that are caused by access path changes. This feature is automated fully with Db2 12 function level (FL) 505 and later, and it is available for both static and dynamic SQL statements. (Before FL V12R1M505, only regression detection is automated, without automatic resolution of the regressions.)

To evaluate the benefits of SQL optimization, we randomly selected 154 dynamic queries from our query performance regression workloads and compared their performance when we ran them with access paths that were not optimized by Db2ZAI against their performance when we ran them with access paths that the Db2 optimizer chose with Db2ZAI models.

### Measurement environment

The target Db2 subsystem used the following environment:

- ▶ An IBM z14 z/OS logical partition (LPAR) with 16 general CPs, four IBM zSystems Integrated Information Processors (zIIPs), and 512 GB of memory
- ▶ z/OS 2.4
- ▶ One Internal Coupling Facility (ICF), one external Coupling Facility (CF), and CFCC level 23, each with three dedicated CPUs
- ▶ An IBM DS8870 direct access storage device (DASD) controller
- ▶ One-way data sharing with Db2 12 at FL 510

The Db2ZAI system used the following environment:

- ▶ An IBM z14 z/OS LPAR with four general CPs and 512 GB of memory
- ▶ One non-data-sharing Db2 12 subsystem at FL 500 hosting metadata for WMLz and Db2ZAI
- ▶ z/OS 2.4
- ▶ A DS8870 DASD controller
- ▶ WMLz 2.4
- ▶ Db2ZAI 1.5.0

### Measurement scenario description

We randomly chose 154 queries from the Db2 for z/OS performance query regression workloads. The queries are all SQL **SELECT** statements with different levels of complexities. They are invoked through the **DSNTEP3** program (an adapted version of **DSNTEP2**), each with a different job name. Before we ran the measurements, the queries all were running on the Db2 target subsystems for a few months, Db2ZAI already trained (completed learning) with th data, and Db2ZAI optimized access paths were available.

As the base measurement, we ran the query jobs after the Db2 subsystem was restarted with ML stopped, which ensured that the SQL statements were all running with access paths without the influence of Db2ZAI.

Then, to measure how the SQL statements perform with the access paths that Db2ZAI helped to choose, we restarted Db2 and started ML, and then ran the same set of query jobs. With ML started, the query SQL statements used the access paths that Db2ZAI helped to optimize.

### Results

From the Db2ZAI UI for SQL optimization dashboard page for all dynamic statements, as shown in Figure 13-1, 14 out of the 154 (9%) dynamic SQL statements improved, and these statements are estimated to save 66% CPU on average.

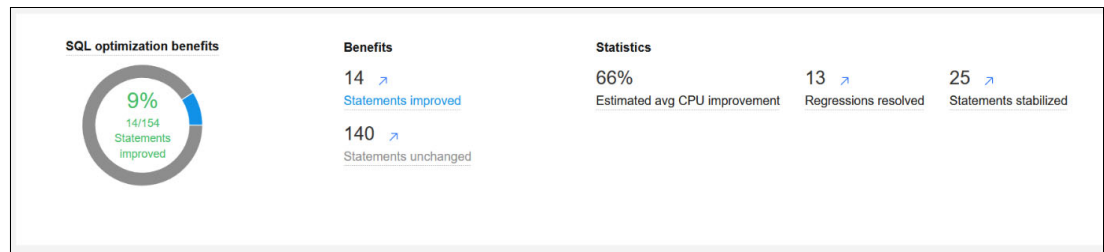


Figure 13-1 Db2ZAI 1.5.0 SQL optimization benefits dashboard: Dynamic statements

By using the accounting traces of the selected dynamic SQL statements, we validated the statements' performance changes and saw that they are in line with what results that are displayed in the Db2ZAI UI.

Figure 13-2 shows the class 2 elapsed time and CPU time of the SQL statements that were run with the access paths that Db2 chose without Db2ZAI models influencing the decisions, and the access paths that the Db2 optimizer chose by using the Db2ZAI models. Most statements show similar performance, although with Db2ZAI, a few statements have notable shorter class 2 elapsed time and CPU time (where the orange line is below the blue line).



Figure 13-2 Dynamic SQLs class2 elapsed time and CPU without versus with Db2ZAI

Figure 13-3 shows the average class 2 elapsed time and class 2 CPU time of the 154 statements, comparing their performance when they were run without Db2ZAI and with Db2ZAI. The average class 2 elapsed time was reduced by 63.6% for the sample SQL statements. The average class 2 CPU time was reduced by 16.8%.

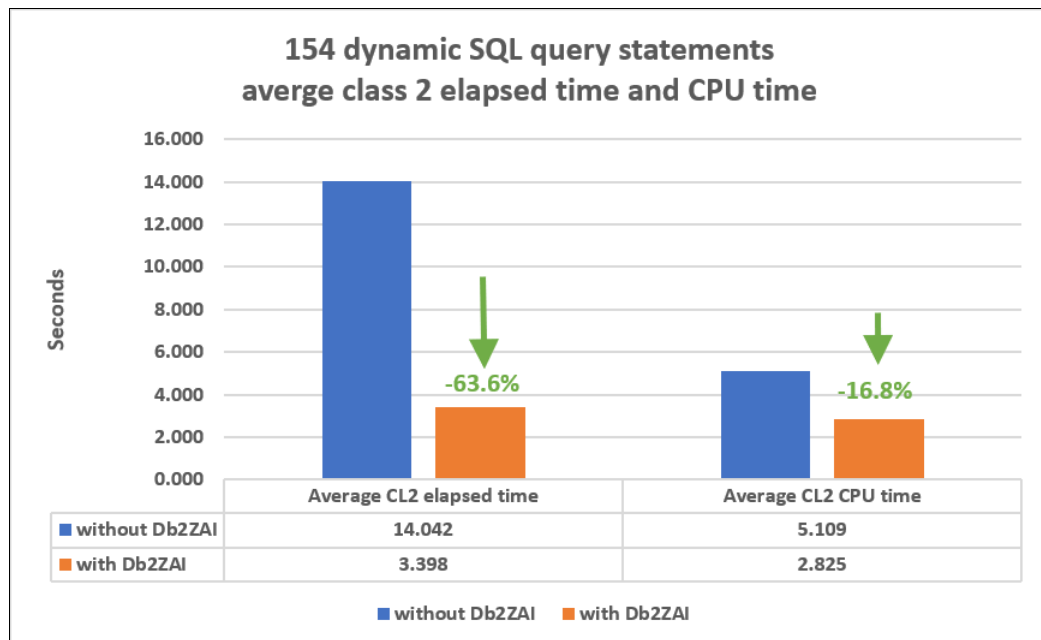


Figure 13-3 Dynamic SQLs average class 2 elapsed time and CPU without versus with Db2ZAI



Figure 13-4 shows the average class 2 elapsed time and class 2 CPU time of the 14 statements that experienced improved performance with access paths that Db2 chose by following the advice that was generated by Db2ZAI. In the SQL optimization dashboard, which is shown in Figure 13-1 on page 367, Db2ZAI estimated the 14 improved statements to have 66% CPU improvement. From the accounting traces of the 14 statements for our measurements, we got 61.8% of class 2 CPU improvement. The estimate that Db2ZAI generated is not far from the actual improvement.

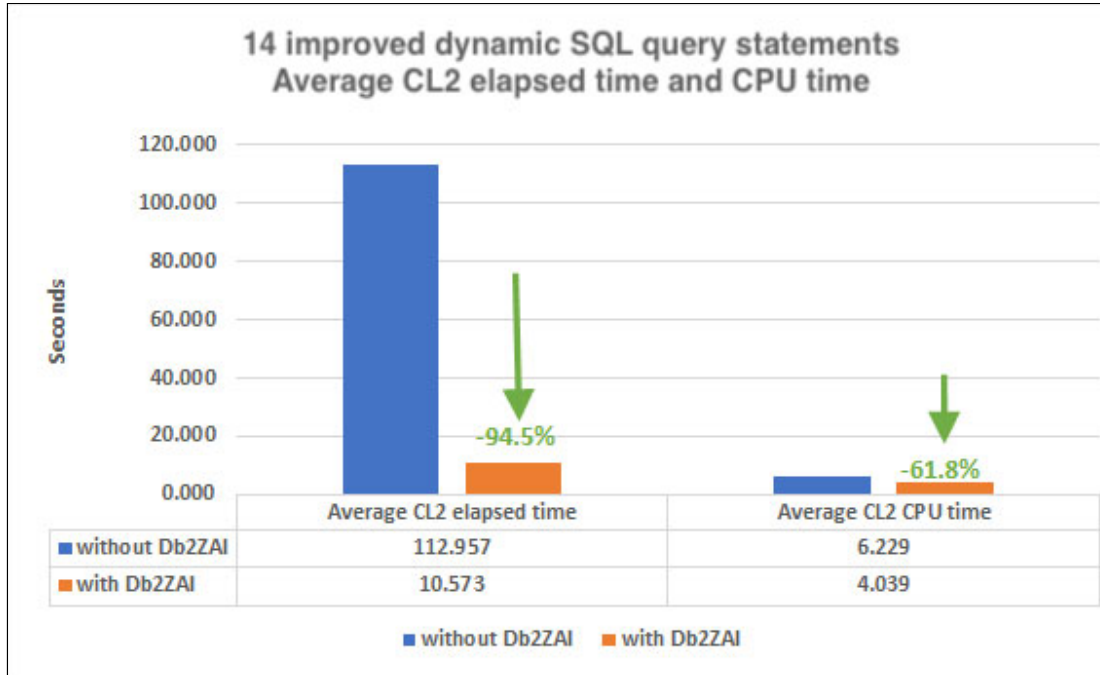


Figure 13-4 Improved SQLs average class 2 elapsed time and CPU without versus with Db2ZAI

Figure 13-5 shows a Db2ZAI UI screen capture of the statement details for a dynamic SQL statement (the batch job name is K01Q0030) with improved performance after implementing the access paths with the Db2ZAI host variable model. Db2ZAI calculated the improvement based on the SQL statement's runtime history, and estimated it to have a 65% CPU improvement and a 94.9% elapsed time improvement.

**Note:** The Statement CPU runtime plot time graph shows only the most recent runtime history records up to 500. In Figure 13-5, the baseline runtime history is not displayed because it was not among the most recent 500 run times.

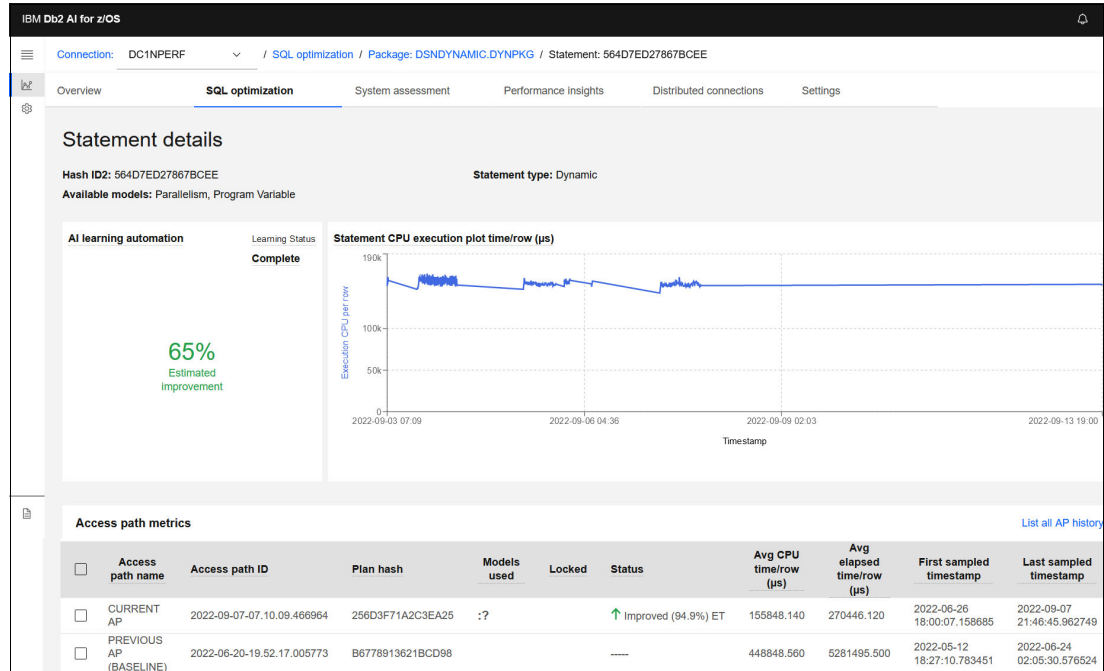


Figure 13-5 Db2ZAI 1.5 statement details for an improved SQL statement

When we examined the accounting trace for the same query, the results of which are shown in Figure 13-6 on page 371, we saw that the class 2 elapsed time and CPU time of the query had improvements of 94.6% and 61.7%, which are similar to what Db2ZAI estimated from the history data of 94.9% and 65%.

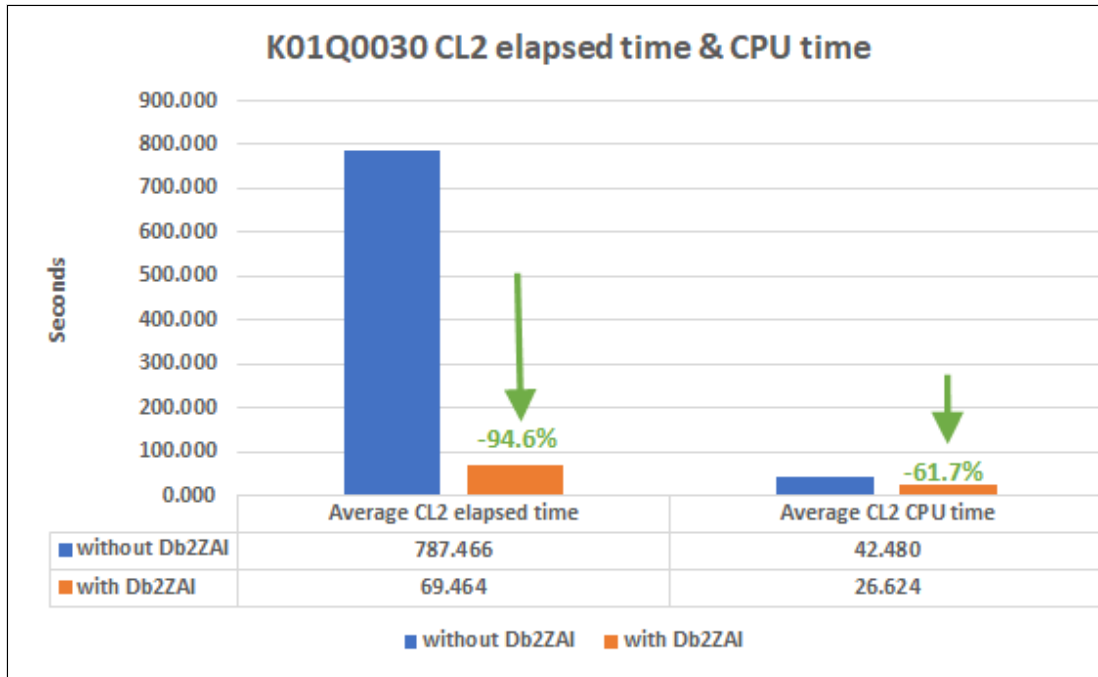


Figure 13-6 K01Q0030 class 2 elapsed and CPU time without Db2ZAI versus with Db2ZAI

### Summary of the results

To summarize, the SQL optimization capabilities of Db2ZAI can improve SQL performance by helping the Db2 optimizer choose better access paths, which means that your applications can benefit from more efficient SQL run times. Db2ZAI manages the baseline data collection, the training, and performance regression monitoring automatically for individual SQL statements with little administration intervention that is required. Dynamic SQL statements can use Db2ZAI models when the training is complete with the next full prepares. For static SQL statements to use Db2ZAI models, the package that contains the SQL statements must be rebound.

For more information on how to use Db2ZAI to optimize SQL statements, see [Optimizing SQL access paths](#).

## 13.2 How Db2ZAI helps with Db2 subsystem management: System assessment and performance insights

The Db2ZAI SA use case aims to offer Db2 system administrators performance tuning recommendations to simplify the challenge of managing Db2 subsystems. The SA use case analyzes the statistics data by processing it through many exception-detection rules. These rules are based on deep domain expertise of Db2 for z/OS professionals in the IBM development labs, and decades of experience gained from Db2 operations at hundreds of customer sites around the world.

In Db2ZAI 1.5.0, a new feature that is called *performance insights* was introduced to provide a quick overview of Db2 health with performance-at-a-glance and deeper drill-down capability for a quick analysis for different types of Db2 connections, CICS, BATCH, DIST, UTILITY, and so on.

This section provides a few examples of how SA and performance insights can help you tune and monitor Db2 subsystem performance. The target Db2 subsystems are a two-way data-sharing group running Db2 12 at FL 510.

### 13.2.1 System assessment and performance insights use case example

Figure 13-7 shows a problem that occurred on 20 September 2022, in which some distributed applications took longer to complete than usual at 00:42, and the slow response situation lasted for approximately 10 minutes. Before Db2ZAI was available, to investigate this performance incident, we had to use a monitoring tool such as IBM OMEGAMON for Db2 Performance Expert on z/OS (OMPE) or a third-party equivalent to print the accounting and statistics reports and try to find clues in many pages of performance numbers. This process was time-consuming, and it was difficult to find any useful information quickly.

However, with the help of performance insights and SA, we were able to identify the application that was causing the problem quickly.

The following steps describe how to pinpoint the “bad” SQL statements:

1. Log in to the Db2ZAI UI and go to the Performance insights page to examine the Db2 Elapsed Time plot under Response Time - Subsystem with the start and end date and time set to between 09/22/2022 00:30 and 09/22/2022 01:00. Because the main symptom of the performance incident is a response time increase for distributed applications, we focus on connection type DIST, as shown in Figure 13-7.

**Note:** Performance insights data is automatically collected after ML starts and Instrumentation Facility Component Identifier (IFCID) 369 traces are turned on. You can check real-time history data with a delay of approximately 10 minutes from the Performance insights of the Db2ZAI UI.

We see that the average response time of DIST transactions is unusually high between 00:40 and 00:50.

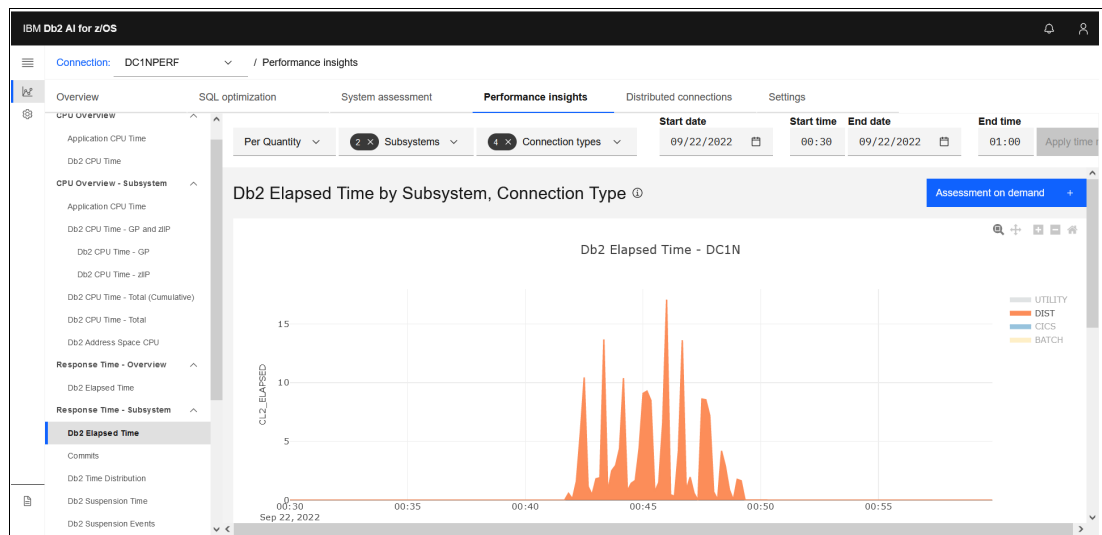


Figure 13-7 Performance insights: Db2 elapsed time for DIST connections - Per Quantity

**Note:** *Per Quantity* means the data points that are shown are the averages that are calculated by dividing the total by the number of accounting record occurrences in each interval.

- To check what might be causing the elapsed time increase, we look at the Db2 Suspension time, and we see that global contention and lock and latch suspensions are the main contributors to the suspension time, as shown in Figure 13-8.

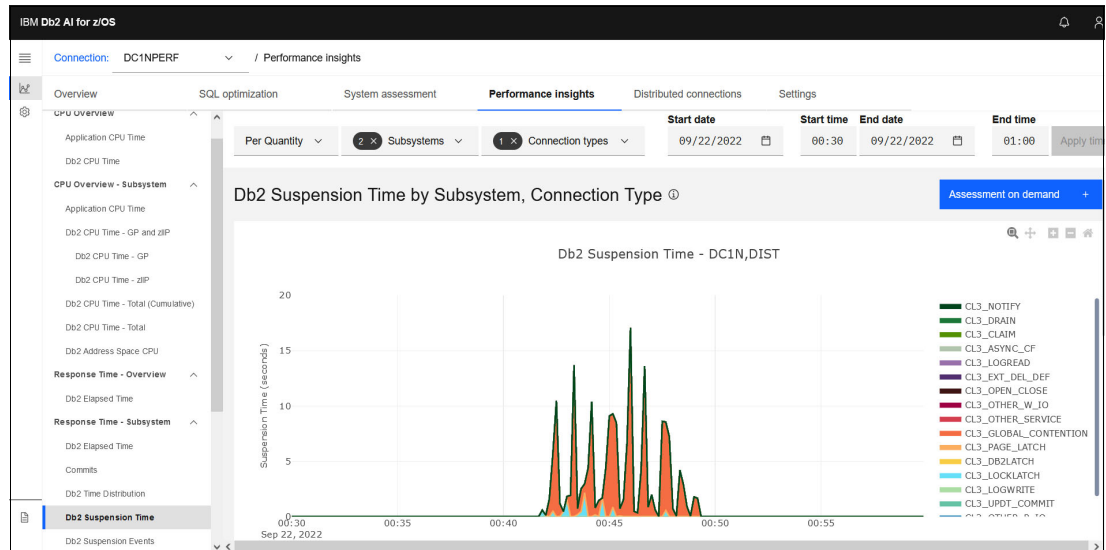


Figure 13-8 Performance insights: Db2 suspension time for DIST connections - Per Quantity

The distributed applications running on our target Db2 subsystems usually run quickly. Why are there such long global contention suspensions?

- We check other connection types and find that BATCH also shows unusually long global contention and lock and latch suspension times at a similar time, as shown in Figure 13-9.

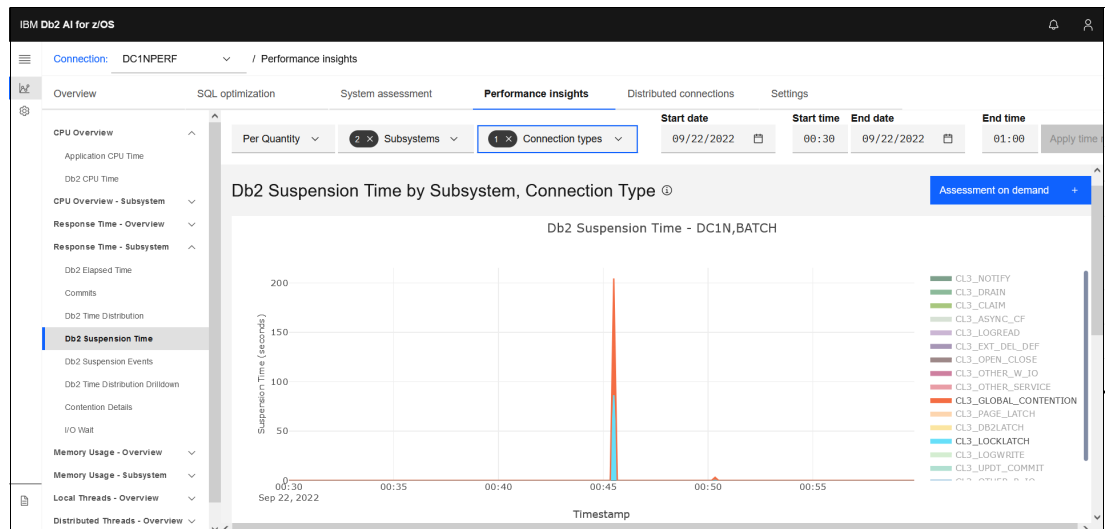


Figure 13-9 Performance insights: Db2 suspension time for BATCH connections - Per Quantity

- We click **Assessment on demand** to trigger a SA with start and end times of 09/22/2022 00:30 and 09/22/2022 01:00. After the on-demand assessment completes, we go to the SA page to see whether there are any exceptions that match our performance issue. We see that the assessment flagged the CL3\_GLOBAL\_CONTENTION\_ALLC metric, which had exceptions between 00:42 and 00:49, as shown in Figure 13-10. The exceptions match what we learned from the performance insights drill-down.

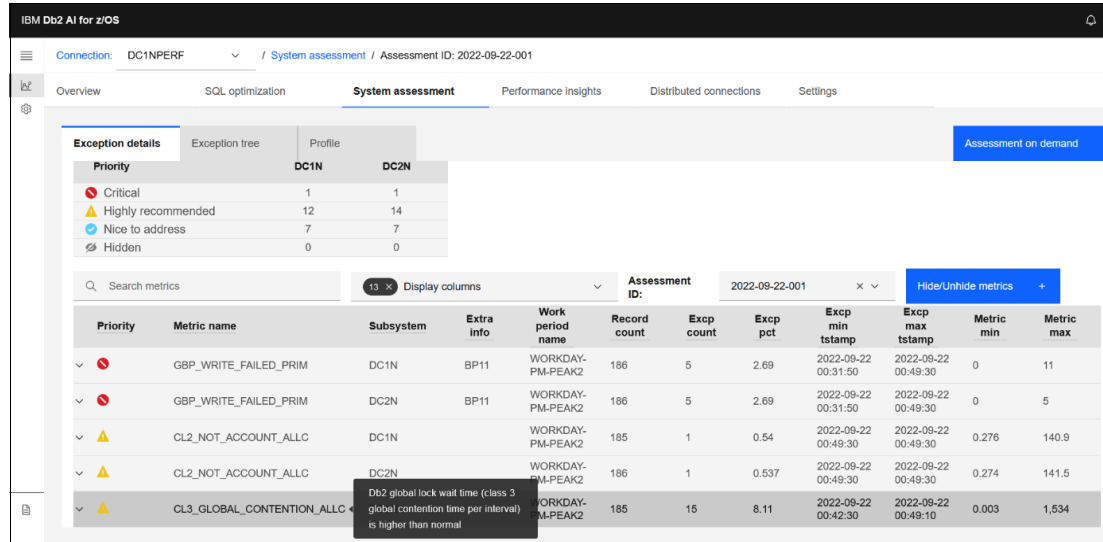


Figure 13-10 System assessment: Exception details

After we open the plots of the CL3\_GLOBAL\_CONTENTION\_ALLC metric, as shown in Figure 13-11, the exceptions are easy to identify: the blue line, which represents CL3\_GLOBAL\_CONTENTION\_ALLC, is going over the dotted red line, which represents the (acceptable) exception threshold.

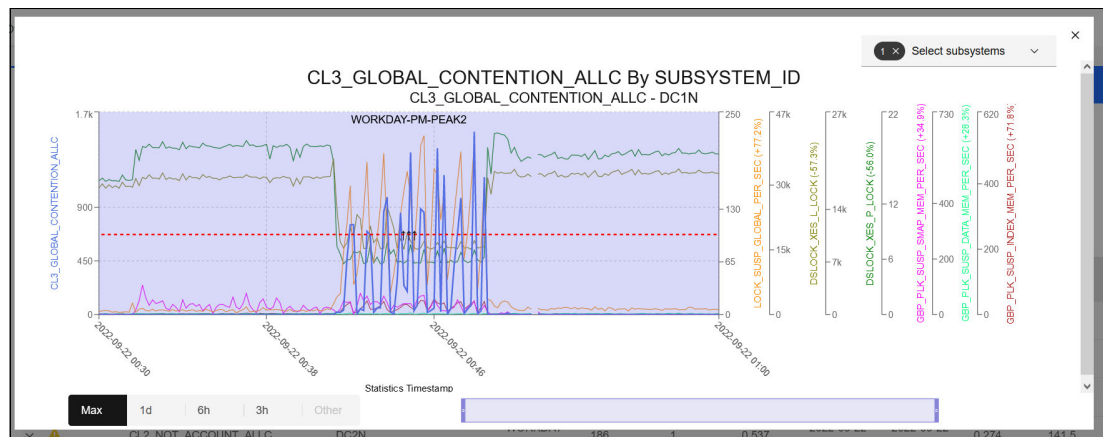


Figure 13-11 System assessment: CL3\_GLOBAL\_CONTENTION\_ALLC

- We want to find which high CPU-consuming SQL statements were running during the abnormal period. Using our cursor, we click and highlight the time range 00:42 - 00:49. Then, we right-click the CL3\_GLOBAL\_CONTENTION\_ALLC graph and click **View top CPU consuming SQLs**, which opens the window that is shown in Figure 13-12 on page 375.

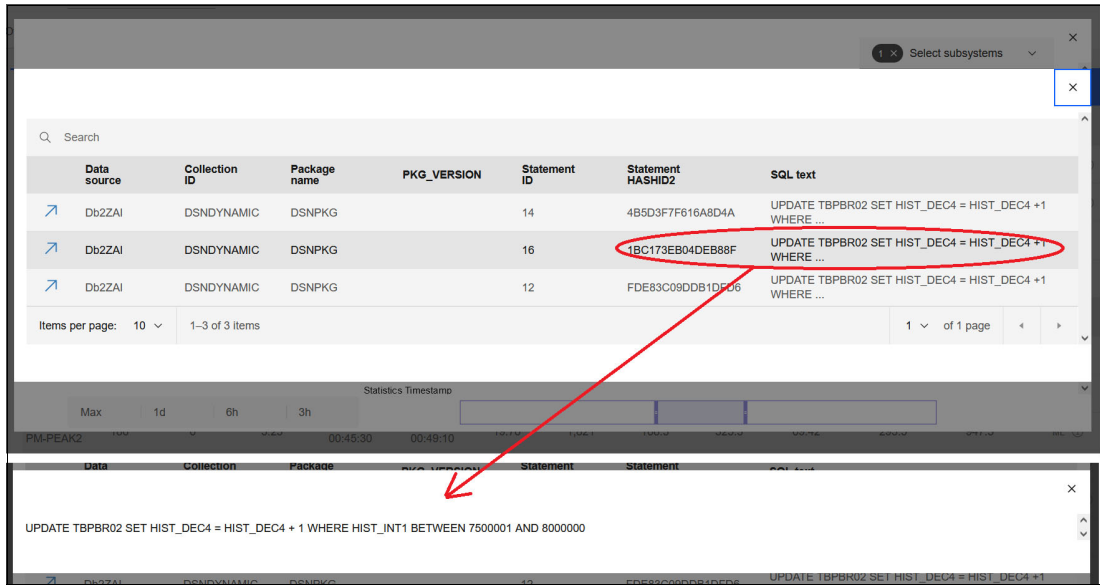


Figure 13-12 System assessment: SQL statements that are consuming the most CPU

We see that three SQL statements are flagged as the top CPU consumers. Clicking the SQL text shows that they are doing updates on table TBPBR02 with large data ranges (in fact, each of the **UPDATE** statements updated 1.5 million rows). With such a large update unit of recovery that locks out so many rows of data for a long period, the unusual global contention of the distributed workload is explained (some of the distributed transactions also update table TBPBR02).

### 13.2.2 System assessment in-depth recommendations

Running Db2 workloads with optimum performance requires much work. There are many performance metrics in the Db2 statistics and accounting data that must be analyzed before any performance tuning opportunities can be identified and implemented. Because of the complexity of such performance studies, they are conducted relatively infrequently. One of the most important missions of the Db2ZAI SA feature is to offer in-depth recommendations about how to respond to the exceptions that are identified by SAs and how to solve them.

Figure 13-13 lists a few of the recommendations that might help you tune your Db2 subsystems without extensive performance studies. These windows are all screen captures from actual SAs.

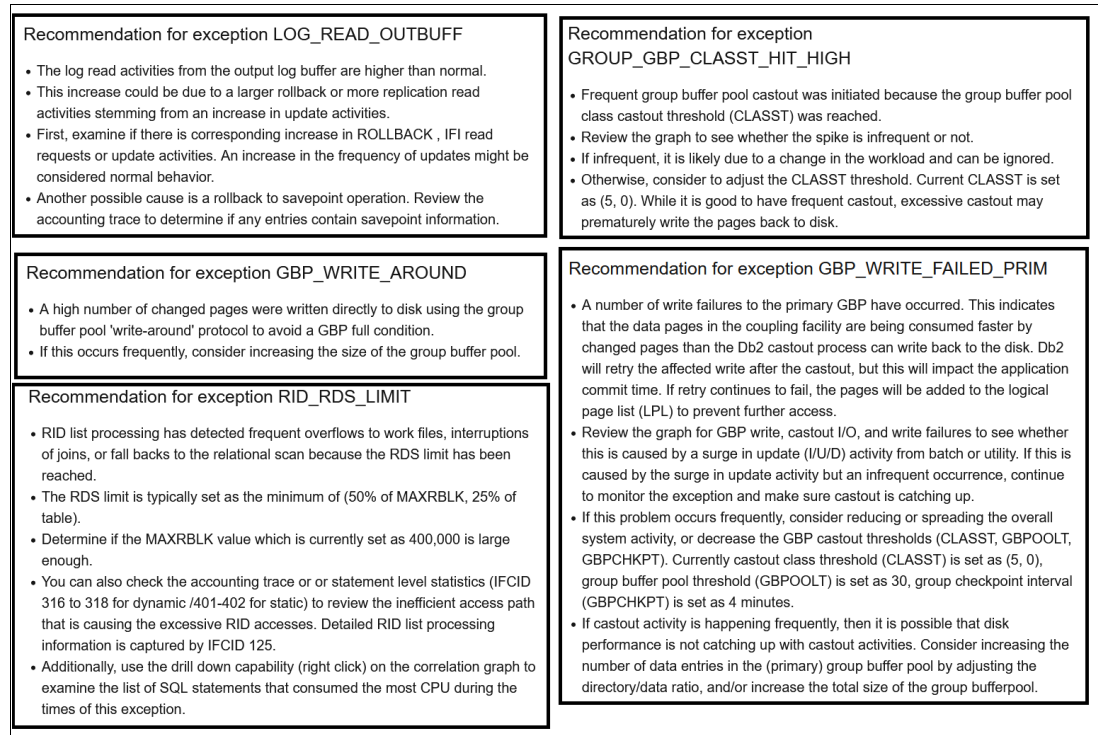


Figure 13-13 System assessment: Recommendations

If you schedule SAs daily, Db2ZAI will help evaluate your Db2 systems' wellness and detect exceptions within 24 hours of them happening, and with minimum human intervention and extra cost. With the provided recommendations, you can decide how to address the exceptions so that your Db2 subsystems run better.

### 13.3 How Db2ZAI helps manage inbound network traffic: DCC

The DCC feature of Db2ZAI deals with distributed connections.

Db2 administrators often lack the knowledge of remote application behaviors and might not be able to implement enough protection for the Db2 subsystems to prevent a flood of distributed connections, which can have a system-level impact in serious cases. A balance is required to control Db2 distributed connections.

The connection controls can be set either at the subsystem level with Db2 subsystem parameters such as MAXDBAT or MAXCONQN, or they can be managed through more granular means such as by using the Db2 profile tables. Setting the connection controls too high can cause Db2 subsystems to be flooded by some rogue applications sending in too many requests in a short period. Setting them too low can prevent the distributed applications from doing their work efficiently enough, so that they are always queuing for available database access threads (DBATs) or being rejected altogether.



Db2ZAI DCC provides a solution to this situation by providing the following capabilities:

- ▶ Automatically learn about distributed connection and thread behavior:
  - In Db2ZAI 1.4, learning is location-based.
  - In Db2ZAI 1.5, application-level controls are introduced, and thread behavior can be learned based on client user IDs or client application names, and the locations from which the remote requests are coming.
- ▶ Identify and prevent problems before they cause an impact.
- ▶ Improve the visibility of your Db2 environment with interactive connection and thread statistics.
- ▶ Use learned behavior to identify why threads are getting blocked.
- ▶ Provide profile recommendations to prevent connection flooding from impacting other applications.

This section provides a few use case examples by using our Db2ZAI 1.5 installation. The target environment is a Db2 12 two-way data-sharing group running at FL 510.

### 13.3.1 Db2ZAI DCC use case: Setting up profiles to monitor and control distributed connections

If you want to implement more granular control over your distributed workloads but find it difficult to gather all the information that you need to define proper profile table entries, the Db2ZAI DCC function will help you. The Db2ZAI product documentation describes a typical flow of how to use the Db2ZAI DCC function to create profile recommendations by training with history statistics data of distributed connections and threads. For more information and Db2ZAI 1.5 instructions, see [Monitoring and controlling distributed connections](#).

This section describes the process. We use some screen captures that show how we set up distributed connection profile monitoring. We use Db2ZAI 1.5 to illustrate this process. The target Db2 subsystems are a two-way data-sharing group running Db2 12 at FL 510.

#### Preparing to monitor distributed connections

With Db2ZAI 1.5, DCC supports client-level profile monitoring (at the client user IDs or client application name level). For client traces, we choose to activate client user ID for our scenario, as shown in Figure 13-14.

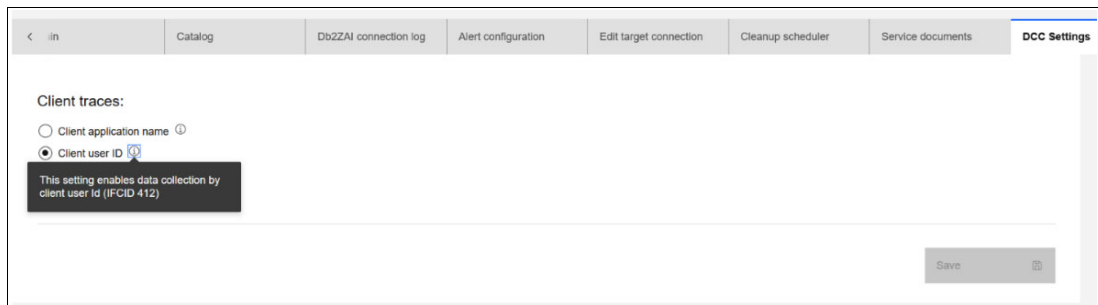


Figure 13-14 DCC settings

Then, we make sure that ML is started on our target Db2 subsystems and that IFCIDs 365, 402, and 412 are active, which ensure that Db2ZAI collects the information that these IFCIDs produce and writes the data into the Db2ZAI tables for DCC training and data presentation.

We set the Db2 `STATIME_MAIN` subsystem parameter to 10 (seconds), and we set the `MAXDBAT` and `CONDBAT` subsystem parameters to 2000 and 10000.

### Setting client priorities

After the distributed connection-related statistics collection is started, Db2ZAI automatically aggregates the statistics based on locations and client user IDs (or client application names, depending on the DCC settings).

To indicate to DCC where to focus, we change the priority levels of the locations and client user IDs from the initial UA (Unaccounted for) to SP (Standard priority), or HP (High priority) for the important ones, before running the training. Assigning priority levels other than UA allows Db2ZAI to generate unique profile recommendations for the locations and client user IDs. Otherwise, they are included in the general profile of location 0.0.0.0.

Figure 13-15 shows how to change the priority levels of the different locations that Db2ZAI identified. Similarly, use the **Client application name** and **Client user ID** tabs to assign priorities to the client application names and user IDs that Db2ZAI identified.

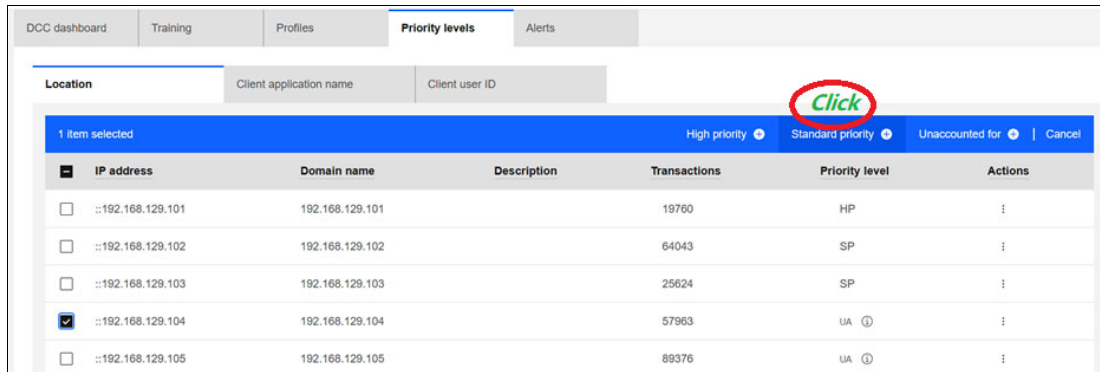


Figure 13-15 Priority levels

### Conducting training and generating profiles

With proper priority levels that are configured for the locations and client user IDs, we run our distributed workloads consecutively for 2 weeks with IFCIDs 365, 402, and 412 active and ML started on the target Db2 subsystems. Then, we train with the 2-week statistics. Db2ZAI DCC training lets Db2ZAI produce profile recommendations based on the training results.

Figure 13-16 shows how to trigger the training.

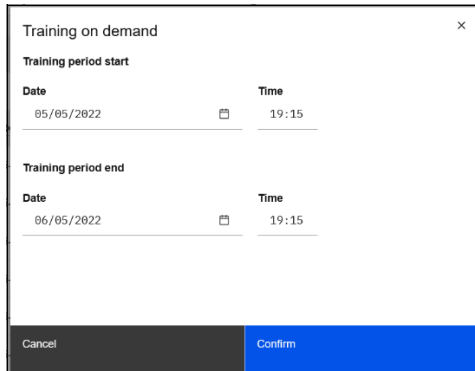


Figure 13-16 DCC: Training on demand

Click **Confirm** to run the DCC training process. The training might take a while to complete, depending on how much statistics data must be processed.

The training completed successfully for our on-demand training, as shown in Figure 13-17.

DCC dashboard		Training	Profiles	Priority levels	Alerts		
<input type="text" value="Search"/> <span style="float: right;"> <input type="text" value="Search"/> <span style="background-color: #0070C0; color: white; padding: 2px 5px;">Training on demand</span> </span>							
Training ID	Training period start	Training period end	Training status	Started by	Profile activation status		Actions
2022-09-25-00.01.53	2022-05-05 19:15:00	2022-06-05 19:15:00	Success	wmizid	<input type="checkbox"/> Warnings inactive <input type="checkbox"/> Exceptions inactive		⋮

Figure 13-17 DCC: Successful training results

**Note:** You can get a training status “Warning” if the sum of high-water mark (HWM) values of all the connection profiles that were generated for a Db2 member exceeds its **CONDBAT** value, or if the sum of HWM values of threads of user ID (or application name) profiles for a Db2 member exceeds its **MAXDBAT** value. In such cases, you must decide whether the warning is acceptable or update the **MAXDBAT** or **CONDBAT** setting and retrain.

### Evaluating recommended profiles

By clicking the training ID, we can check the warning and exception threshold recommendations for each location (connection thresholds) and client user ID (thread thresholds). We also can change the connection and thread warning and exception thresholds when necessary.

As shown in Figure 13-18, the location with IP address 192.168.129.101 had a connection HWM value of 80 on member DC1N. The profile recommendations on the connection warning and exception thresholds for this location on member DC1N are 81 and 123.

With client user ID traces active, the training produces profile recommendations for warning and exception thresholds at the thread level for each client user ID. (If the client application name is chosen from the DCC settings, the recommendations are for each client application name). In the following example with a thread HWM of 106 on DC1N, Db2ZAI recommends that the USRT001 profile use warning and exception thresholds of 107 and 287.

Overview	SQL optimization	System assessment	Performance insights	Distributed										
<table border="1"> <thead> <tr> <th>IP Address</th> <th>Domain</th> <th>Subsystem</th> <th>Connection HWM</th> <th>Threshold</th> </tr> </thead> <tbody> <tr> <td>192.168.129.101</td> <td>192.168.129.101</td> <td>DC1N</td> <td>80</td> <td>51</td> </tr> </tbody> </table>					IP Address	Domain	Subsystem	Connection HWM	Threshold	192.168.129.101	192.168.129.101	DC1N	80	51
IP Address	Domain	Subsystem	Connection HWM	Threshold										
192.168.129.101	192.168.129.101	DC1N	80	51										
<table border="1"> <thead> <tr> <th>Attribute thresholds</th> <th>Current values</th> <th>Recommended values</th> </tr> </thead> <tbody> <tr> <td>Connection warning threshold</td> <td>---</td> <td>81</td> </tr> <tr> <td>Connection exception threshold</td> <td>---</td> <td>123</td> </tr> </tbody> </table>					Attribute thresholds	Current values	Recommended values	Connection warning threshold	---	81	Connection exception threshold	---	123	
Attribute thresholds	Current values	Recommended values												
Connection warning threshold	---	81												
Connection exception threshold	---	123												
<table border="1"> <thead> <tr> <th>IP Address</th> <th>Domain</th> <th>Subsystem</th> <th>Connection HWM</th> <th>Threshold</th> </tr> </thead> <tbody> <tr> <td>192.168.129.101</td> <td>192.168.129.101</td> <td>DC2N</td> <td>121</td> <td>100</td> </tr> </tbody> </table>					IP Address	Domain	Subsystem	Connection HWM	Threshold	192.168.129.101	192.168.129.101	DC2N	121	100
IP Address	Domain	Subsystem	Connection HWM	Threshold										
192.168.129.101	192.168.129.101	DC2N	121	100										
<table border="1"> <thead> <tr> <th>Attribute thresholds</th> <th>Current values</th> <th>Recommended values</th> </tr> </thead> <tbody> <tr> <td>Connection warning threshold</td> <td>---</td> <td>122</td> </tr> <tr> <td>Connection exception threshold</td> <td>---</td> <td>186</td> </tr> </tbody> </table>					Attribute thresholds	Current values	Recommended values	Connection warning threshold	---	122	Connection exception threshold	---	186	
Attribute thresholds	Current values	Recommended values												
Connection warning threshold	---	122												
Connection exception threshold	---	186												

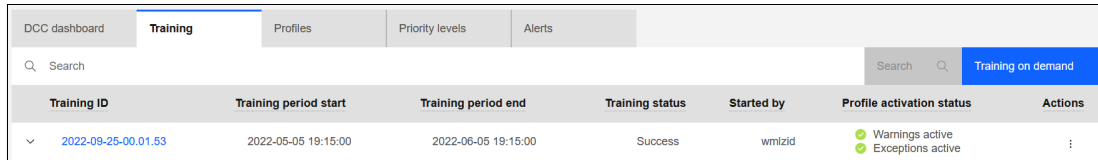
  

Overview	SQL optimization	System assessment	Performance insights	Distributed												
<table border="1"> <thead> <tr> <th>User ID</th> <th>Domain</th> <th>Subsystem</th> <th>Thread HWM</th> </tr> </thead> <tbody> <tr> <td>USRT001</td> <td>DC1N</td> <td></td> <td>106</td> </tr> </tbody> </table>					User ID	Domain	Subsystem	Thread HWM	USRT001	DC1N		106				
User ID	Domain	Subsystem	Thread HWM													
USRT001	DC1N		106													
<table border="1"> <thead> <tr> <th>Attribute thresholds</th> <th>Current values</th> <th>Recommended values</th> </tr> </thead> <tbody> <tr> <td>Thread warning threshold</td> <td>---</td> <td>107</td> </tr> <tr> <td>Thread exception threshold</td> <td>---</td> <td>287</td> </tr> <tr> <td>Queue depth</td> <td>NULL</td> <td>NULL</td> </tr> </tbody> </table>					Attribute thresholds	Current values	Recommended values	Thread warning threshold	---	107	Thread exception threshold	---	287	Queue depth	NULL	NULL
Attribute thresholds	Current values	Recommended values														
Thread warning threshold	---	107														
Thread exception threshold	---	287														
Queue depth	NULL	NULL														
<table border="1"> <thead> <tr> <th>User ID</th> <th>Domain</th> <th>Subsystem</th> <th>Thread HWM</th> </tr> </thead> <tbody> <tr> <td>USRT001</td> <td>DC2N</td> <td></td> <td>120</td> </tr> </tbody> </table>					User ID	Domain	Subsystem	Thread HWM	USRT001	DC2N		120				
User ID	Domain	Subsystem	Thread HWM													
USRT001	DC2N		120													
<table border="1"> <thead> <tr> <th>Attribute thresholds</th> <th>Current values</th> <th>Recommended values</th> </tr> </thead> <tbody> <tr> <td>Thread warning threshold</td> <td>---</td> <td>121</td> </tr> <tr> <td>Thread exception threshold</td> <td>---</td> <td>325</td> </tr> <tr> <td>Queue depth</td> <td>NULL</td> <td>NULL</td> </tr> </tbody> </table>					Attribute thresholds	Current values	Recommended values	Thread warning threshold	---	121	Thread exception threshold	---	325	Queue depth	NULL	NULL
Attribute thresholds	Current values	Recommended values														
Thread warning threshold	---	121														
Thread exception threshold	---	325														
Queue depth	NULL	NULL														

Figure 13-18 DCC: Training and profile recommendations

## Activating recommended profiles

After evaluating the profile items and changing them as needed, we activate the profiles. Figure 13-19 shows the green check boxes that indicate that both profile warnings and exceptions are active.



Training ID	Training period start	Training period end	Training status	Started by	Profile activation status	Actions
2022-09-25-00.01.53	2022-05-05 19:15:00	2022-06-05 19:15:00	Success	wmlzid	Warnings active Exceptions active	

Figure 13-19 DCC: Training, active warning, and exception profiles

We are ready to let the profiles do their work and monitor distributed work coming in from the monitored locations and running under the monitored client user IDs.

**Note:** If your workload has major changes after you complete the profile evaluation and activation process (such as new applications being introduced, work coming in from new locations, or peak workload increases), you must retrain and repeat the profile evaluation and activation process.

### 13.3.2 Db2ZAI 1.5 DCC use case: Visualizing distributed workload activities with the dashboard

Db2ZAI 1.5 introduced a dashboard for DCC that serves as an entry point for Db2 administrators to get quickly an idea about how the distributed workloads are running on the target Db2 systems.

The dashboard is the first page that you are presented with after opening the Distributed connections web page from the Db2ZAI web UI.

This section provides some examples of what the dashboard looks like and how you can go to different sections of the dashboard to discover more information about your distributed workloads.

The dashboard is split into three sections:

- ▶ DBAT High Water Mark
- ▶ Distributed connections and thread usage metrics
- ▶ Profile exceptions and warning alerts

#### DBAT High Water Mark

The DBAT High Water Mark dashboard, which is shown in Figure 13-20 on page 381, shows the DBAT statistics of the last 24 hours for all the target Db2 subsystems (under the current Db2ZAI connection) in 15-minute intervals.

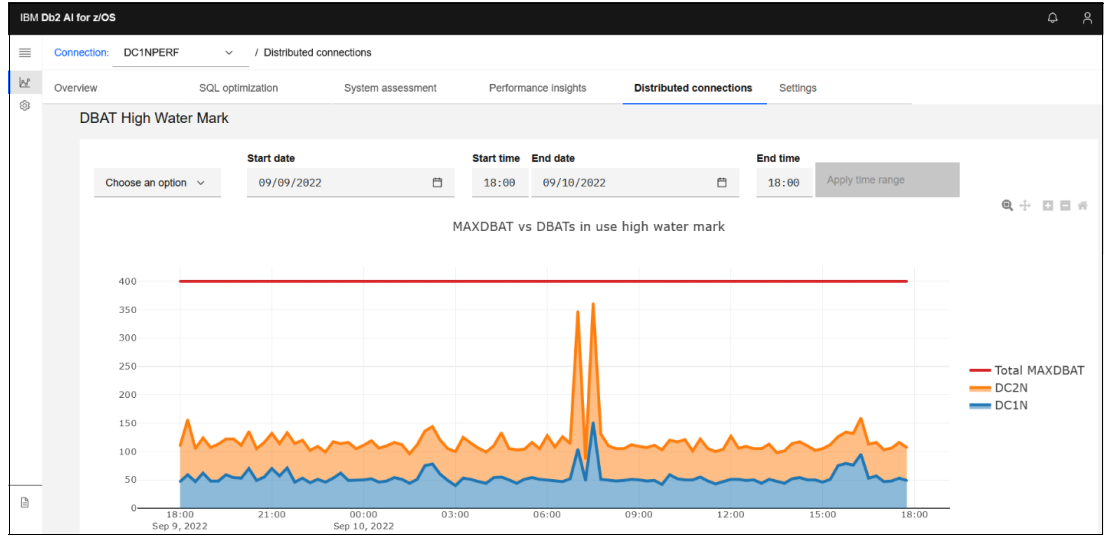


Figure 13-20 DCC dashboard: DBAT High Water Mark

From here, you can right-click to focus on a single Db2 member, as shown in Figure 13-21, or right-click again to investigate thread-level statistics by either location, user ID, or application name.

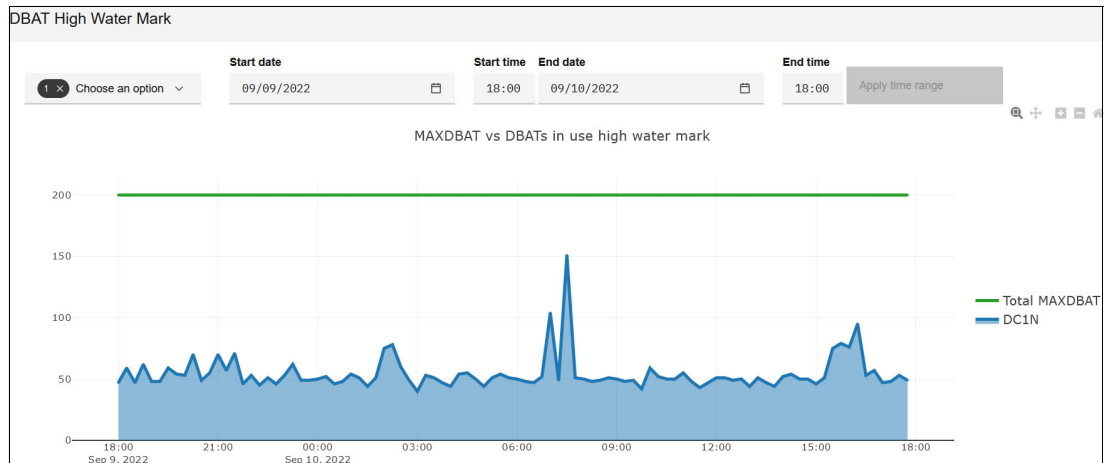


Figure 13-21 DCC dashboard: DBAT High Water Mark for a single Db2 member

Click **User ID Threads** to open a graph of the top 20 user IDs with the largest number of threads. As shown in Figure 13-22, clicking a user ID in the graph shows the thread statistics of the specific user ID, where you can select different metrics to view.



Figure 13-22 DCC dashboard: DBAT high water mark and drilling down to thread statistics

## Distributed connections and thread usage metrics

The second part of the dashboard shows you how the distributed workloads are running on your target Db2 subsystems (data-sharing group level) in four dimensions:

- ▶ How many of the connections are running with sysplex workload balancing (WLB)?
- ▶ How many of the connections are secured with SSL?
- ▶ How many of the distributed transactions are terminating normally?
- ▶ How many of the distributed transactions are running with thread pooling?

These factors are important ones that can help DBAs determine whether the distributed workloads still have room for improvement. For example, as shown in Figure 13-23, 71.6% of the connections (the time range that is covered is the same as specified in the DBAT High Water Mark section) used sysplex WLB. This result means that 28.4% of the connections are not using WLB, and they are vulnerable to situations such as network failures, member maintenance, and reaching the MAXDBAT threshold, which affect the Db2 member to which they connect.

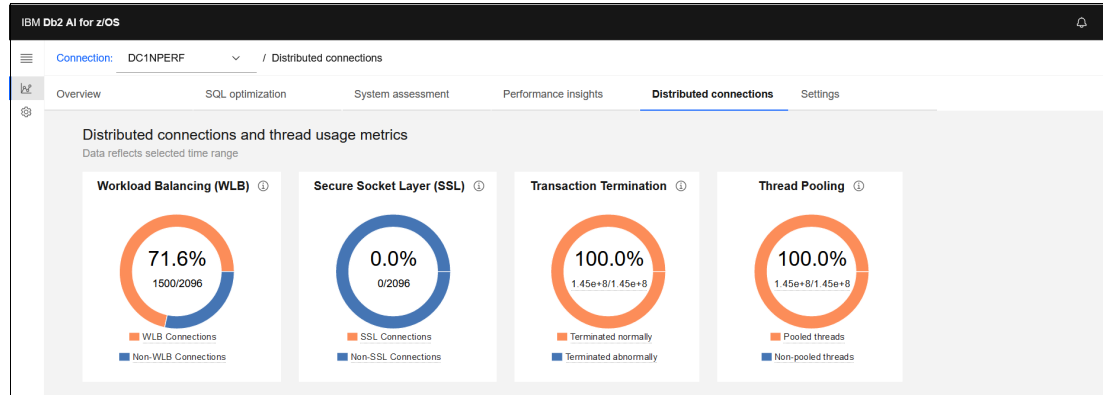


Figure 13-23 DCC dashboard: Distributed connections and thread usage metrics

Clicking the WLB circle displays the top 20 locations that are not using sysplex WLB, as shown in Figure 13-24. We can click each of the locations to check their connection statistics.

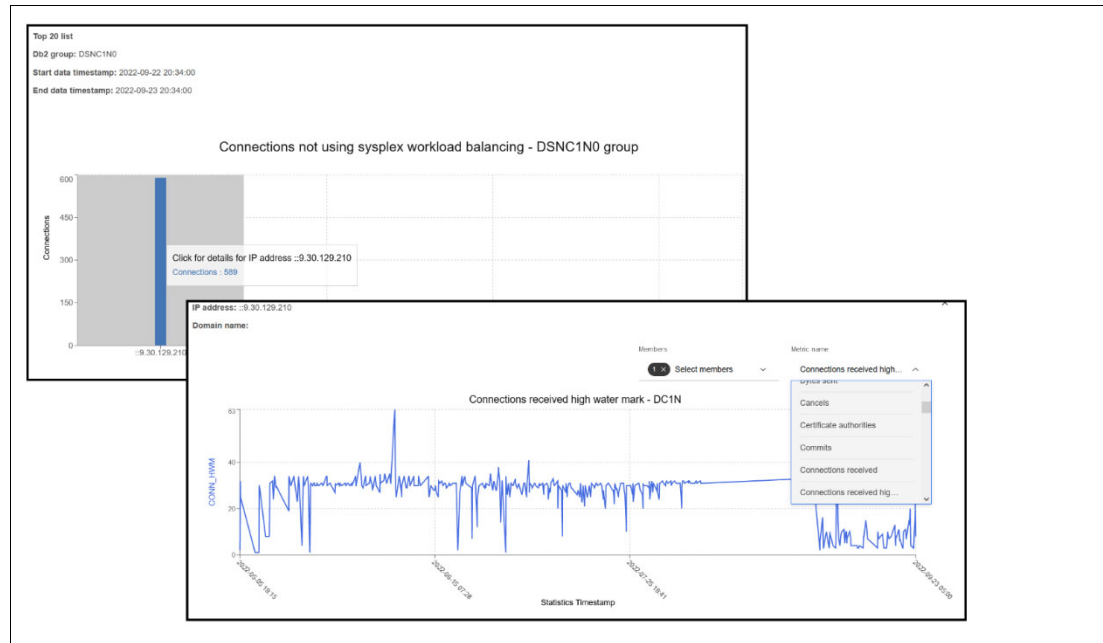


Figure 13-24 DCC dashboard: WLB drill-down to connection statistics

## Profile exceptions and warning alerts

The third section of the dashboard shows a summary of the DCC profile exception and warning alerts for the last 7 days.

As shown in Figure 13-25, we can do a drill-down analysis from the dashboard to get to the list of alerts, and then display the alert details web page where we can check various connection and thread statistics scorecards for the monitored resources.

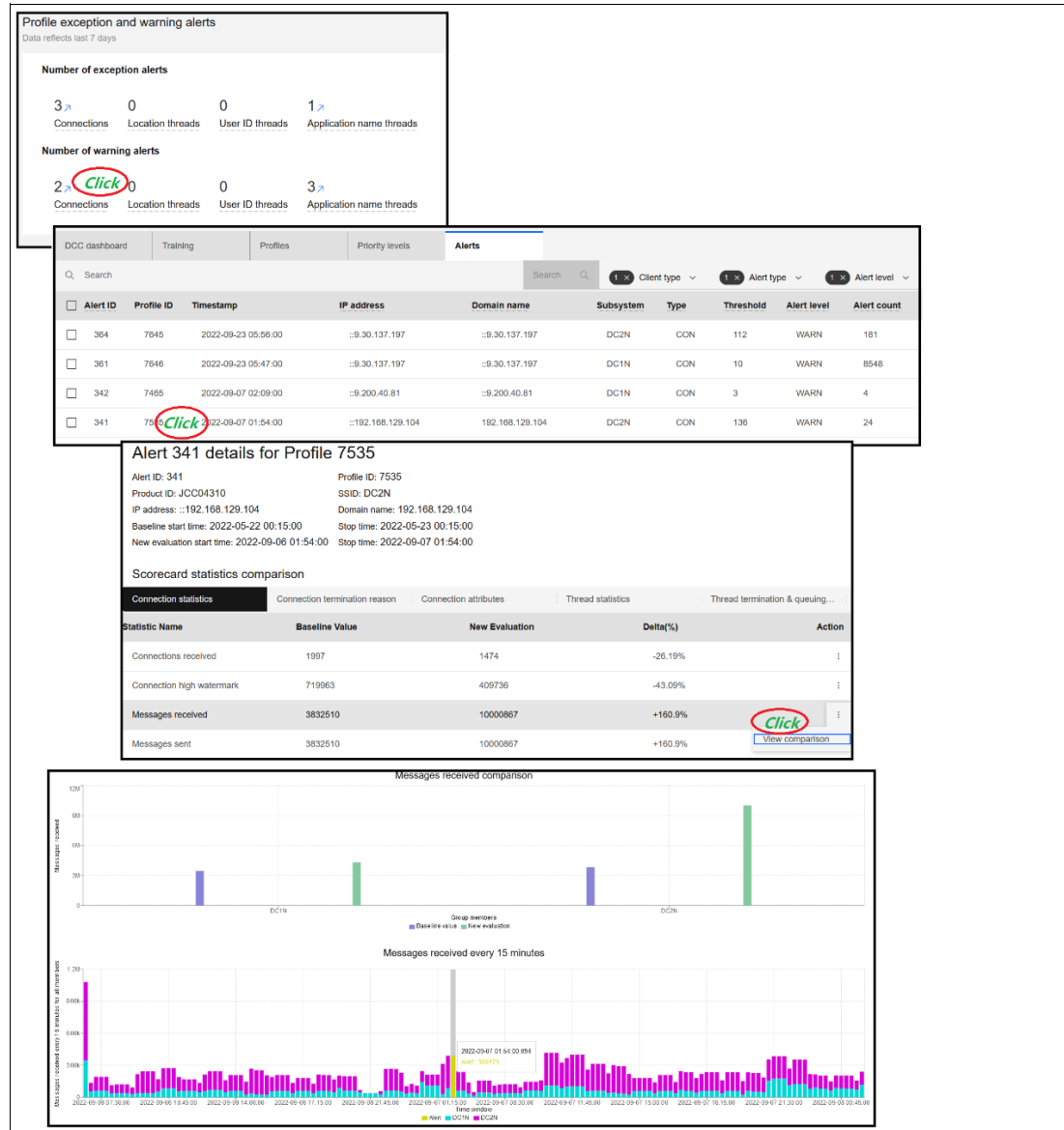


Figure 13-25 DCC dashboard: Profile exception and warning alerts drill down

Drilling down into the profile exceptions and warning alerts can help DBAs determine whether the alert is due to something erratic happening or if the workload has grown over time and the profile must be adjusted or retrained to accommodate workload growth.



## 13.4 Summary of capacity requirements and general recommendations

This section describes where the Db2ZAI processes that can consume system resources take place and offers some general recommendations about how to monitor Db2ZAI resource consumption.

Figure 13-25 on page 384 is a high-level depiction of the Db2ZAI architecture, which reflects both the Db2ZAI 1.4 and 1.5 designs.

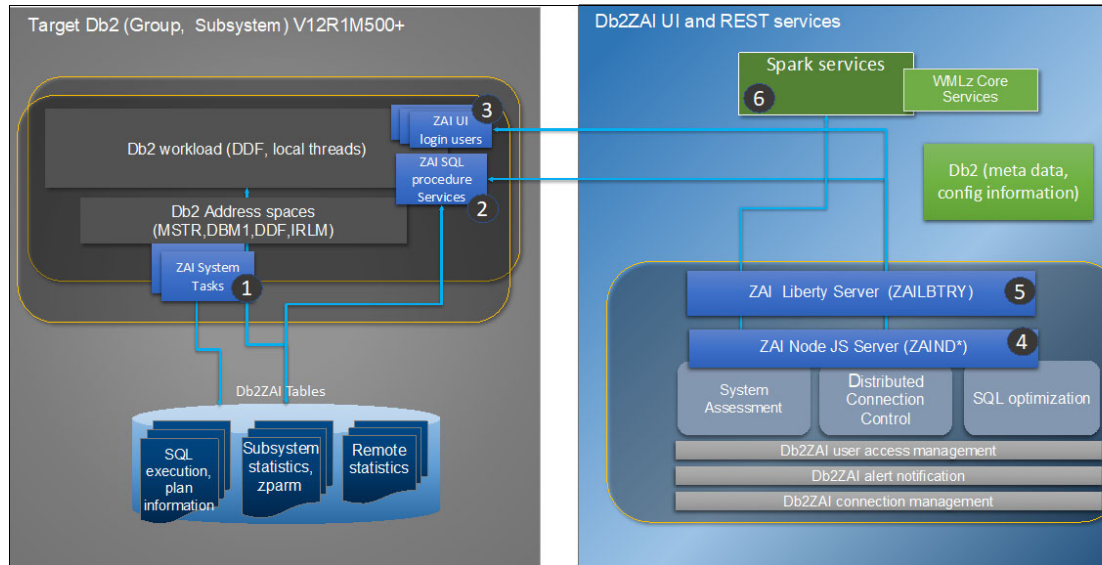


Figure 13-26 Db2ZAI high-level architecture

This architectural diagram shows the six areas where Db2ZAI consumes system resources to fulfill its functions. The following list explains the function and the CPU cost that is associated with each of these six areas and provides some general recommendations for managing and monitoring them through z/OS Workload Manager (WLM) report and service classes.

The first three areas are on the target Db2 subsystems:

- Db2ZAI system tasks are responsible for base data collection and SQL sampling and monitoring. Base data collection includes inserting and updating Db2ZAI tables to gather Db2 subsystem statistics, local and group buffer pool (GBP) statistics, SQL run histories, distributed connection and thread statistics, and so on. All these tasks run in the *ssnmDBM1* address space and are dispatched as 100% zIIP eligible preemptive System Recovery Boosts (SRBs).

The recommended setup is to use a separate WLM report class for *ssnmDBM1* address spaces for better monitoring of the Db2ZAI IIP processing time.

- Data aggregation processes run periodically to aggregate the data that is collected, such as the SQL run histories and statistics data collected by the SA and DCC. The data aggregation processes run mostly as SQL stored procedures. Other than DCC data aggregations, most of them are called by the Db2ZAI system tasks and run in the *ssnmDBM1* address space with their CPU cost 100% zIIP eligible. DCC aggregations are triggered from the Db2ZAI UI and run as Distributed Relational Database Architecture (DRDA) requests, whose CPU time is 60% zIIP eligible.

- ▶ When Db2ZAI users log on to the UI and invoke UI operations, the data access of the Db2ZAI tables is performed at the primary target Db2 member to which the Db2ZAI is connected. They are DRDA requests whose CPU cost is 60% zIIP eligible, and they run under the user's UI login user ID.

The recommended setup is to use a separate WLM report class and possibly a separate service class for DDF enclave SRB processing for the user IDs that perform Db2ZAI UI operations.

The next three areas are on the Db2ZAI UI and REST services side:

- ▶ Db2ZAI uses Node.js to support the UI web interfaces. The default names of the Node.js address spaces are ZAIND\*. They are UNIX System Services (OMVS) processes that consume general central processor (CP) CPU time.

The recommended setup is to use a dedicated WLM report class for these address spaces for better monitoring of their resource consumption.

- ▶ The main Db2ZAI infrastructure functions run within an IBM WebSphere® Liberty server. The Liberty server runs under UNIX System Services, and its default address space name is ZAILBTY. The Liberty services are zIIP eligible. During SAs, the Liberty server can consume up to 2.7 zIIP processors in our measurements with STATIME\_MAIN=10 specified for the target Db2 systems.

The recommended setup is to use a dedicated WLM service class for the Liberty address space and assign it an importance lower than other more critical tasks that demand zIIP capacity on the LPAR.

- ▶ SA training runs under z/OS Spark services (the default address space names are ALNSPK\* for WMLz 2.4) through WMLz (the default address space names are ALN\*). The z/OS Spark services are zIIP eligible. Because SA training should be infrequent, the CPU demand for the z/OS Spark services is not high.

The recommended setup is to use WLM report classes for the WLMz and Spark address spaces to monitor the training cost. If frequent SA training is expected, consider setting up a separate service class to control zIIP usage.

**Note:** The Db2 subsystem that is used to store Db2ZAI metadata is not reflected in the architecture depiction. Db2ZAI accesses the metadata through DRDA requests only occasionally. The system resource requirement for hosting the Db2ZAI metadata is small and can be ignored.

Figure 13-27 on page 387 shows some examples of WLM service and report classes for the various processes and address spaces that are involved in an Db2ZAI installation.

Target Db2 : example of WLM class setup				
----Qualifier---		-----Class---		
Type	Name	Service	Report	
STC	TN	DB01DBM1	DB2ADDR	DB01DBM1
STC	TN	DB01MSTR	DB2ADDR	DB01MSTR
DDF	UI	ZAIUSR*	DDFWORK	ZAIUSR

Db2ZI UI and Rest services : example of WLM class setup				
----Qualifier---		-----Class---		
Type	Name	Service	Report	
OMVS	TN	ALNSPK*	OMVSWML	SPRKWORK
OMVS	TN	ALN*	OMVSWML	WMLZWORK
OMVS	TN	SPARK*	OMVSSPK	SPRKWORK
OMVS	TN	ZAIND*	ZAINODE	ZAINODE
OMVS	TN	ZAILBTY	ZAILBTY	ZAIWORK

Figure 13-27 Db2ZAI related WLM service and report class examples

**Note:** For the WMLz, z/OS Spark, and the Db2ZAI Liberty and Node.js address spaces, if they are started as started tasks, use subsystem type STC to assign their WLM service and report classes.

Disk requirements on the target Db2 systems side are mainly for the Db2ZAI tables. Plan for 200 GB of available disk space for them. We consumed 120 GB disk space for the Db2ZAI 1.5 installation without triggering any Db2ZAI table cleanups with a two-way data-sharing target Db2 system. A best practice is to monitor your disk usage for the Db2ZAI objects and to set up proper retention periods for different types of Db2ZAI data, as described in [Scheduling Db2ZAI table cleanups](#). Doing a periodical online **REORG** of the Db2ZAI objects is also a best practice.

Disk requirements on the Db2ZAI UI side are for the product installation and the working directories for the Db2ZAI services. Plan for 20 GB of disk space for Db2ZAI usage, which is also enough to cover the disk space that WMLz 2.4 requires to support Db2ZAI.

## 13.5 Detailed capacity evaluations for Db2ZAI 1.5

To determine the capacity requirements for running Db2ZAI 1.5 on both the target Db2 side and the Db2ZAI UI side, we created a mixed workload to feed Db2ZAI the data that it needs for all the major functions.

This section provides the capacity evaluation test results from our in-house measurements for Db2ZAI 1.5 running with Db2 12 target subsystems that host the mixed workload.

### Measurement environment

The target Db2 subsystem consisted of the following setup:

- ▶ One IBM z14 LPAR with 16 general CPs, four zIIPs, and 512 GB of memory
- ▶ z/OS 2.4
- ▶ One ICF, one external CF, and CFCC level 23, each with three dedicated CPUs
- ▶ An IBM DS8870 disk controller

- ▶ One Linux on IBM Z LPAR with six general CPs and 24 GB of memory to drive DRDA workloads
- ▶ Two-way data sharing, at Db2 12 FL 510, with both members running on the same LPAR

The Db2ZAI UI system consisted of the following setup:

- ▶ One IBM z14 LPAR with four general CPs and 512 GB of memory
- ▶ One non-data-sharing Db2 12 subsystem at FL 500 hosting metadata for WMLz and Db2ZAI
- ▶ z/OS 2.4
- ▶ A DS8870 disk controller
- ▶ WMLz 2.4
- ▶ Db2ZAI 1.5.0

### Measurement scenario description

We created a mixed workload to feed Db2ZAI the data that it needs for all the major functions to determine the capacity requirements for running Db2ZAI 1.5 on both the target Db2 side and the Db2ZAI UI side.

We defined separate WLM report classes for the address spaces and DDF tasks that are involved, and we used RMF workload activity reports to obtain the CPU usage and real storage usage performance numbers. On the target Db2 side, we also used OMPE reports for some of the Db2 accounting and statistics numbers.

The background workload that runs on the target Db2 subsystems is made up of the following subworkloads:

- ▶ Dynamic and static SQL statements that are invoked through batch jobs.
- ▶ A subset of DRDA IBM Relational Warehouse Workload (IRWW) transactions:
  - One set running with 40 different IP addresses
  - Another set running with 40 different user IDs and 40 different application names
- ▶ A random update batch workload.
- ▶ A two-way data-sharing relational transaction workload (RTW) CICS-Db2 workload that adds more complexity to the system performance insights drill-downs.

Because we expect more clients to start running Db2 subsystems with STATIME\_MAIN=10, which is the default for Db2 13 for z/OS, we ran our tests with both target Db2 members with STATIME\_MAIN=10.

Before conducting the measurements, we accumulated Db2ZAI data for over 6 weeks without cleanup. The Db2ZAI tables take up about 120 GB of disk space. There were about 2000 SQL statements in scope (meaning Db2ZAI monitors their performance and analyzes their behavior for AI access path optimization). At the time of the measurements, most of the in-scope SQL statements were in “Learning complete” status, which means that Db2ZAI evaluated their access paths and created proper ML models for them if beneficial.

We evaluated the system resource cost for four major types of Db2ZAI usage scenarios:

- ▶ Basic Db2ZAI data collection and aggregations
- ▶ Performing UI operations
- ▶ SA training
- ▶ SAs

Some configuration and workload factors affect how much data Db2ZAI collects and how much data the various Db2ZAI functions must process. The data volume affects the cost of running Db2ZAI. The following factors affect Db2ZAI performance:

- ▶ The number of target Db2 members
- ▶ SQL optimization factors:
  - The number of SQL statements in scope and their runtime frequencies
  - The number of packages in scope
  - SQL statement complexities (related to IFCID 318 overhead at the target Db2 subsystem)
  - For dynamic statements, the full prepare occurrences
- ▶ SA and performance insights factors:
  - The **STATIME\_MAIN** subsystem parameter setting
  - The number of allocated buffer pools and GBPs
  - The number of work periods
- ▶ DCC factors:
  - The **STATIME\_MAIN** subsystem parameter setting
  - The number of IP addresses, client user IDs, and application names

Here are the characteristics of the configuration that we used during the capacity requirement studies for Db2ZAI 1.5.

- ▶ Number of Db2 members: 2
- ▶ Number of SQL statements in scope: ~2000
- ▶ Number of packages in scope: ~1000
- ▶ SQL complexities: All complexity levels
- ▶ Number of full prepares: 0
- ▶ Number of local BPs per Db2 member: 34
- ▶ **STATIME\_MAIN** setting: 10
- ▶ Number of group BPs per Db2 member: 16
- ▶ Number of IP addresses: 42
- ▶ Number of work periods per Db2 member: 13
- ▶ Number of remote applications: 47
- ▶ Number of remote client IDs: 41
- ▶ Number of Db2 connection types: 3
- ▶ Number of Db2ZAI table rows added per hour: ~550,000

We conducted the following measurements to understand the costs that are associated with Db2ZAI processing:

- ▶ We ran the background workload for 60 minutes after we issued the **-STOP ML** command and used this run as the base measurement for evaluating Db2ZAI data collection and aggregation cost. We call this measurement *NOML*.
- ▶ We ran the background workload again for 60 minutes after we issued the **-START ML** command. We made sure that no Db2ZAI UI operations were performed during the measurement period. We call this measurement *NOUI*. The performance differences between the NOUI and NOML measurements reflect the Db2ZAI data collection and aggregation cost.

- ▶ We ran the third measurement with the background workload running for 60 minutes after we issued the **-START ML** command, and concurrently we performed a typical set of various Db2ZAI UI operations. We call this measurement *UI*. The performance differences between the UI and NOUI measurements should reflect the Db2ZAI UI operation cost.
- ▶ For evaluating SA training performance, we ran two SA training processes while the background workload ran with ML started. During this time, we ensured that no other Db2ZAI UI operations were performed. The first training (measurement TR1) was with 1 week of the statistics data that Db2ZAI. The second training (measurement TR2) was with 2 weeks of statistics data. By comparing TR1 and TR2 data with NOUI performance data, we can understand how systems assessment training performs and the training performance scales as the data volume increases.
- ▶ To evaluate SA performance, we ran three SAs while the background workload ran with ML started and with no other Db2ZAI UI operations. The three assessments assessed 1 day, 2 days, and 3 days of data, and they are assigned measurement IDs of SA1, SA2, and SA3. Comparing the SA1-3 performance data with NOUI, we calculated how SAs perform and how the assessment performance scaled as the data volume increased.

Table 13-1 summarizes the measurements.

*Table 13-1 Db2ZAI 1.5 performance evaluation measurement characteristics*

Measurement ID	Background workload running?	ML started?	Db2ZAI UI operations performed?	Description
NOML	Yes	No	No	Base for NOUI.
NOUI	Yes	Yes	No	Compare to NOML to understand the data collection and aggregation cost. Base for UI, TR1-2, and SA1-3.
UI	Yes	Yes	Yes	Compare to NOUI to understand the typical UI operation performance.
TR1 and TR2	Yes	Yes	No	Compare to NOUI to understand the SA training performance.
SA1, SA2, and SA3	Yes	Yes	No	Compare to NOUI to understand the SA performance.

### 13.5.1 High-level capacity evaluation results

This section presents the high-level observations from our Db2ZAI 1.5 capacity evaluations. Detailed performance numbers are provided in the following sections.

The CPU cost of each scenario is expressed as the percentage of a single CPU's capacity that is used during the measurement periods. To illustrate the results, look at the numbers for basic Db2ZAI data collection and aggregation.

The CPU cost that is spent on general CP is approximately 36 seconds, and approximately 140 seconds on zIIP processors for the measurement duration of 60 minutes. The CPU cost on general CP is calculated as  $36 / (60 \times 60) = 1\%$  of a single CPU's capacity. Similarly, the CPU cost on zIIP processors is calculated as  $140 / (60 \times 60) = 2.9\%$ . Therefore, the total CPU cost is  $1\% + 2.9\% = 3.9\%$  of the capacity of a single processor.

Table 13-2 lists the CPU cost for our capacity evaluation tests with our specific setup and data volume. Different configurations most likely see a different CPU cost for the various Db2ZAI functions. However, if you see a much bigger and out-of-proportion CPU cost than what is listed in Table 13-2, you should contact IBM to determine whether the cost is reasonable.

Table 13-2 Db2ZAI 1.5 capacity evaluation: High-level overview of CPU cost

Db2ZAI functions	Processing on	CPU cost (% of a single CPU's capacity)	How much of the CPU cost is zIIP eligible?	Main processing under
Basic Db2ZAI data collection and aggregations	Target Db2 side	3.9%	> 70%	ssnmDBM1
	Db2ZAI UI side	Ignorable	N/A	N/A
Performing UI operations	Target Db2 side	3.5%	> 50%	DDF requests under UI user ID
	Db2ZAI UI side	1.3%	> 25%	ZAIND* and ZAILBTY
System assessment training 1-week data	Target Db2 side	3.7%	> 60%	DDF requests under Db2ZAI connection scheduler ID
	Db2ZAI UI side	163.7%	> 97%	ALNSP*
System assessment training 2-week data	Target Db2 side	4.8%	> 65%	DDF requests under Db2ZAI connection scheduler ID
	Db2ZAI UI side	131.5%	> 97%	ALNSP*
System assessments 1-day data	Target Db2 side	2.0%	> 58%	DDF requests under Db2ZAI connection scheduler ID
	Db2ZAI UI side	240.1%	> 99%	ZAILBTY
System assessments 2-day data	Target Db2 side	3.0%	> 58%	DDF requests under Db2ZAI connection scheduler ID
	Db2ZAI UI side	261.2%	> 99%	ZAILBTY
System assessments 3-day data	Target Db2 side	3.6%	> 58%	DDF requests under Db2ZAI connection scheduler ID
	Db2ZAI UI side	269.8	> 99%	ZAILBTY

**Note:** The *idling cost* of the Db2ZAI services on the LPAR where the WMLz, Node.js, and the Liberty servers are running is approximately 2% of a single CPU (less than 72 seconds of CPU time per hour). *Idling* means that the services are started but are not processing real work.

For the memory usage, we looked at the RMF paging activity reports and used the CENTRAL STORAGE FRAMES counts in “Frame and Slot Counts” section to calculate the maximum real storage that was used during the measurement periods. We used the following equation:

$$\text{maximum\_memory\_used\_in\_GB} = (\text{CENTRAL STORAGE FRAMES TOTAL} - \text{CENTRAL STORAGE FRAMES AVAILABLE (MIN)}) \times 4 / 1024 / 1024$$

As shown in Table 13-3, for our measurements, only SA training used a significant amount of real storage to run, which is 8 GB. Other Db2ZAI functions did not use a significant amount of real storage.

Table 13-3 Db2ZAI 1.5 capacity evaluation: High-level overview of memory usage

Db2ZAI functions	Processing on	Memory usage
Db2ZAI related services up but not doing real work	Db2ZAI UI side	4 GB
Data collection and aggregation	Target Db2 side	Not significant
	Db2ZAI UI side	Not significant
UI operations	Target Db2 side	Not significant
	Db2ZAI UI side	Not significant
SA training	Target Db2 side	Not significant
	Db2ZAI UI side	8 GB
SA assessment	Target Db2 side	Not significant
	Db2ZAI UI side	Not significant

**Note:** The idling real storage cost of the Db2ZAI services on the LPAR where the WMLz, Node.js, and the Liberty servers are running is approximately 4 GB. Idling means that the services are started but are not processing real work.

With the information that is in Table 13-2 on page 391 and Table 13-3, you can see that supporting Db2ZAI is not expensive. With our configuration on the LPAR where the target Db2 subsystems are, the maximum of CPU capacity that we added with all the Db2ZAI functions should be less than 15% of a single processor. On the LPAR where Db2ZAI and WMLz services run, less than 5% of the CPU capacity of a single processor is enough to handle functions other than SA training and assessments. To support SA training, up to two IIP processors might be required during the training. For SAs, up to three IIP processors might be required while the assessments run.



Regarding the usage of real storage with our configuration, we noted the following key findings:

- ▶ On the LPAR where the Db2ZAI services run, approximately 4 GB of real storage is needed for all the services to be started (including WMLz services). An extra 8 GB of real storage is used on the LPAR during SA training processes. All other Db2ZAI functions do not require a significant amount of extra real storage. However, you might want to set aside 15 GB of real storage.
- ▶ We did not observe a significant amount of extra real storage that was consumed for Db2ZAI on the LPAR on where the target Db2 systems ran.

### 13.5.2 Detailed capacity evaluation results

This section presents the detailed performance numbers for the four types of capacity evaluation measurements that we ran.

#### Basic Db2ZAI data collection and aggregations

We ran the background workload for 1 hour without ML running as the base measurement. The dynamic SQL statements during the measurements were running with Db2ZAI optimized access paths.

Then, we ran the workload again after issuing the `-START ML` command on the target Db2 subsystem. During this 1-hour measurement period, we did not perform any UI operations on the Db2ZAI UI. By comparing this measurement with the base measurement, we can see that the difference in CPU and memory usage reflects the Db2ZAI data collection and data aggregation cost.

Based on our analysis, which is shown in Figure 13-28, we made the following general observations about the Db2ZAI 1.5 data collection and the aggregation overhead that was incurred with our configurations.

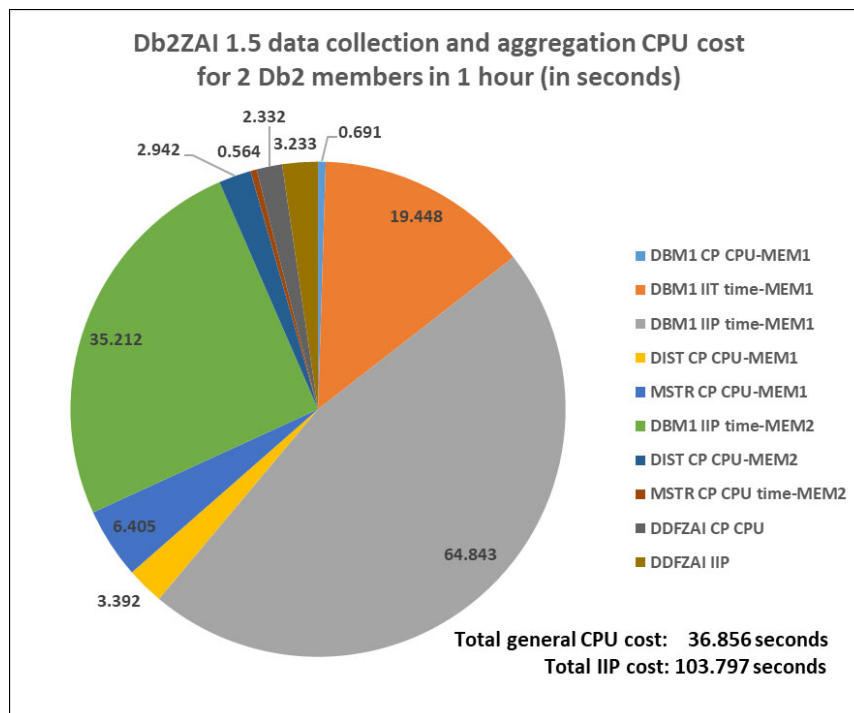


Figure 13-28 Db2ZAI 1.5 data collection and aggregation CPU usage on target Db2 subsystems

- ▶ The CPU cost occurs mainly on the target Db2 side.
  - Figure 13-28 on page 393 shows that a total of approximately 140 seconds of CPU time was used to perform Db2ZAI data collection and data aggregations during our 1-hour measurement period, and 104 seconds of the CPU were spent on zIIP processors, which is 74% of the total CPU cost.
  - Most of the direct data collection and aggregation CPU is zIIP eligible and charged as *ssnmDBM1* SRB time.
  - DCC data aggregation runs as DRDA requests to the primary Db2 member (the Db2 member to which Db2ZAI services connect). The DRDA requests run under the Db2ZAI scheduler ID. The total DCC data aggregation CPU cost was only a few seconds during the 60-minute measurement run.
  - We observed an increase in the *ssnmDBM1* I/O interrupt time on the Db2 member to which Db2ZAI services connect. The increase is attributed to the data handling on the Db2ZAI tables, such as data prefetch activities and synchronous I/Os.
  - The CPU cost for Db2ZAI 1.5 data collection and aggregations in our measurement uses less than 3% of a single CPU's processing time, and over 70% of the CPU cost can be offloaded to zIIPs.
  - To evaluate real storage usage, we examined both the Db2 statistics reports and RMF reports, which showed no noticeable real storage usage difference between the two measurements. RMF paging activity reports show real storage usage of approximately 117 GB on the LPAR where the target Db2 subsystems run.
- ▶ The Db2ZAI UI side spends only a few seconds of CPU to maintain connectivity with the Db2 target systems and trigger DCC data aggregation activities.

## Performing UI operations

To measure the cost of Db2ZAI UI operations, we ran our background workload for 1 hour with ML started on the Db2 target subsystems without any UI operations that were performed from the Db2ZAI UI. This measurement is used as the base measurement.

Then, we ran the background workload again with ML started while we performed all the major Db2ZAI 1.5 UI operations for 1 hour, including typical UI operations for SA, performance insights, SQL optimization, and DCC scenarios. In total, the test required approximately 400 mouse clicks. The performance differences between this measurement and the base measurement should reflect the cost of performing Db2ZAI 1.5 UI operations.

Based on our analysis, we made the following general observations about Db2ZAI 1.5 UI operation system resource consumption:

- ▶ On the Db2 target system side:
  - DRDA requests are sent to the Db2 member that Db2ZAI directly connects to for the UI operations that must retrieve data from the Db2ZAI tables. Db2ZAI code is designed for efficiency and avoids repeated data retrieval, and it uses efficient queries that return data quickly with optimum access paths.
  - Figure 13-29 on page 395 shows the class 2 elapsed time distribution of the DRDA requests for the UI user ID. For 1 hour of Db2ZAI UI operations, only 52.2 seconds were spent for general CPU time and 74.8 seconds were spent on the zIIP processors. These measurements have a total of 3.5% of a single CPU's capacity on average for the 60-minute measurement period.
  - Approximately 400 mouse clicks generated approximately 1400 DRDA transactions with a total class 2 elapsed time of 224.4 seconds, which averages to 0.16 second of class 2 elapsed time per DRDA transaction.

- We observed over 5 seconds of class 3 notify message and other service suspension time, which was due to DCC profile activation in which Db2 uses notify messages to communicate profile activation states between Db2 members.
- To evaluate real storage usage, we examined both the Db2 statistics and RMF reports on the LPAR where the target Db2 subsystems are. They show no noticeable real storage usage difference between the base measurement and the measurement for the UI operations. From the RMF paging activity reports, the real storage usage of the LPAR is approximately 117 GB with a variation within a +-1% range.

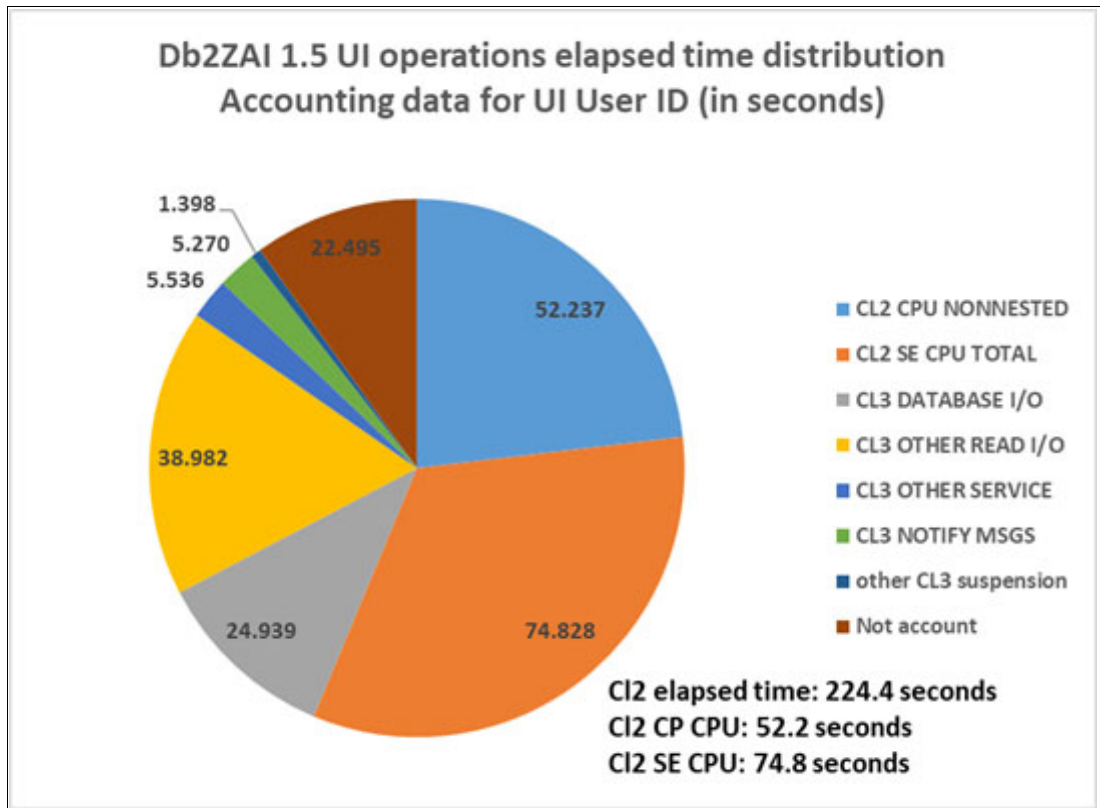


Figure 13-29 Db2ZAI 1.5 UI operations elapsed time and CPU usage for UI user ID on the target Db2

**Note:** A best practice is to use dedicated buffer pools for the Db2ZAI objects. This approach makes monitoring Db2ZAI object access efficiency easier and enables you to tune the buffer pool attributes more easily.

- ▶ On the Db2ZAI UI side:
  - The Node.js address spaces (ZAIND\*) and the Liberty applications handle the web interactions and send DRDA requests to the target Db2 subsystem to retrieve data.
  - The CPU time that is spent in the Node.js address spaces is not zIIP eligible, but the CPU that is spent in the Liberty server is almost 100% zIIP eligible. Figure 13-30 depicts the CPU cost that is incurred on the LPAR where Db2ZAI is installed and running during our UI operations measurement of 1 hour.
  - In our configuration, the total CPU cost is small (no more than 2% of the capacity of a processor).
  - There was no real storage usage increase on the LPAR that hosts the Db2ZAI services when performing UI operations. Both the base measurement and the measurement for the UI operations have a similar LPAR real storage usage of approximately 27 GB.

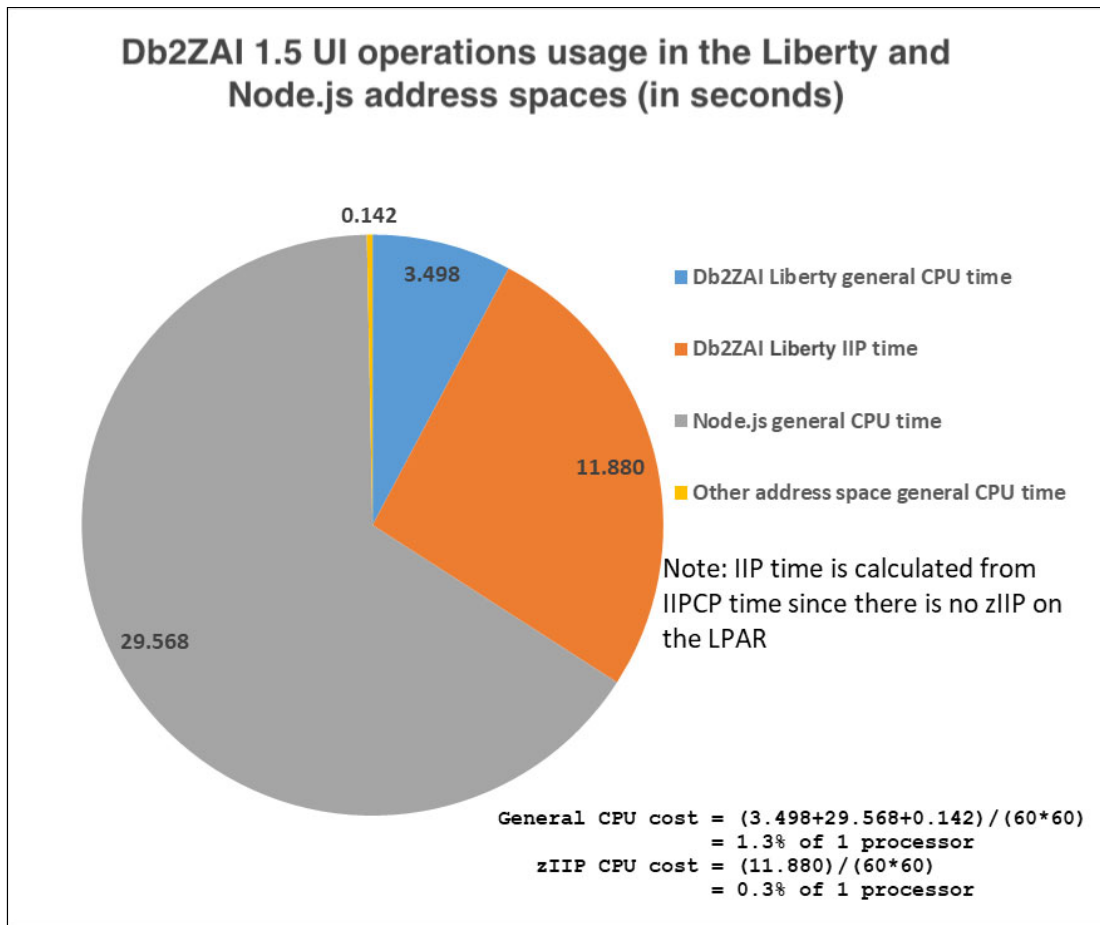


Figure 13-30 Db2ZAI 1.5 UI operations: CPU usage on the Db2ZAI UI side

### System assessment training

SA training helps Db2ZAI establish baseline thresholds on system and workload performance metrics for the target Db2 subsystems. Db2ZAI uses the learned thresholds to identify exceptions that might need database administrator's interventions. For a training process to be effective, the data that is used must cover all the work periods of all the Db2 subsystems that the Db2ZAI connection manages.

We ran the SA training with 7 days and 14 days worth of Db2ZAI history data that was generated with STATIME\_MAIN=10 for the two-way data-sharing group that runs the mixed workload that is described in “Measurement scenario description” on page 367.

Table 13-4 lists some of the important factors that affect training performance.

Table 13-4 Db2ZAI 1.5 system assessment training study: Numbers that affect training performance

For each Db2 member				Data volume	
STATIME_MAIN	Number of distinct BPs	Number of distinct GBPs	Number of work periods	Number of days	Number of rows retrieved from DSN_ML_AKIRA_METRICS_MASTER
10	36	12	24	7	120839
10	36	12	24	14	236857

We configured the Spark applications to run with SPARK\_WORKER\_MEMORY=16G, SPARK\_WORKER\_CORES=2, and executor\_memory=8G, as described in [Configuring WML for z/OS base](#).

By analyzing the performance data from the two SA training processes with different data volume, we made the following observations:

- ▶ On the Db2 target system side:
  - On the target Db2 member to which Db2ZAI directly connects, DRDA requests are sent from the Db2ZAI services to retrieve the data for training and for writing the training results back to the SA tables.
  - As shown in Figure 13-31, the DRDA requests that are related to the SA training processes use a small amount of CPU and zIIP processing time. We see an increase in *ssnmDBM1* IIP SRB time (the pre-emptible SRB time spent on zIIP) as more data is pre-fetched for the training test with 2 weeks of data. As a result, class 3 other read suspension time for the 2-week data training test also increases compared to the 1-week data training. The 4 K and 16 K buffer pools that we use for the Db2ZAI objects have a VPSIZE of 80000 and 20000. These buffer pools can be tuned to be larger for better buffer pool hit ratios to improve the efficiency of training data prefetch.
  - All the extra CPU that is used on the target Db2 subsystems for the SA training process represents less than 4% of a zIIP processor and less than 2% of a general processor engine.
  - There is no real storage usage increase on the LPAR where the target Db2 subsystems run during the SA training tests.

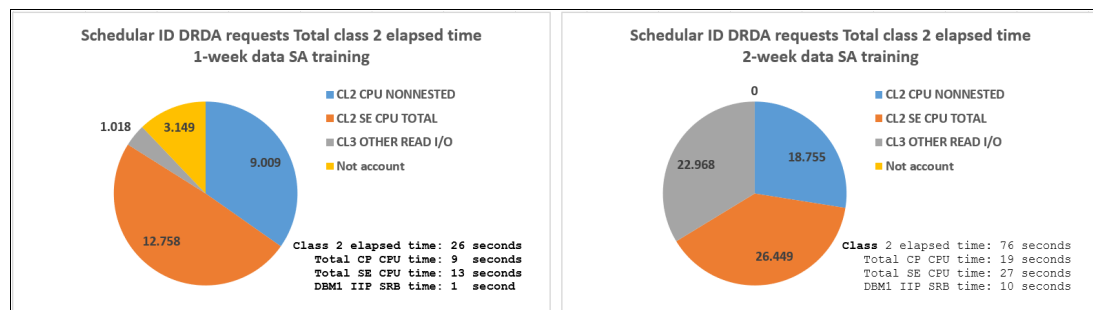


Figure 13-31 Db2ZAI 1.5 SA training elapsed time and CPU usage for scheduler ID on the target Db2

- ▶ On the Db2ZAI UI side:
    - The CPU cost for running the SA training processes is incurred under the z/OS Spark address spaces. The default address space names for Spark are ALNSP\* with WMLz 2.4 installations. Approximately 98% of the CPU time that is spent by Spark address spaces is zIIP eligible. The SA training function scales well as the data volume increases in terms of elapsed time and CPU usage.
    - Figure 13-32 shows the detailed performance numbers for our tests. The “Total LPAR CPU time” is the total CPU that was used by all address spaces running on the LPAR for the durations of the SA training processes. “Spark CPU time” is the CPU time that all the spark address spaces consumed during the SA training processes. Similarly, “Total LPAR IIP time” and “Spark IIP time” show the IIP time that was consumed by all the address spaces on the LPAR and by Spark address spaces during the SA training processes. Because there are no zIIPs that are configured on the LPAR where the Db2ZAI instance runs, we calculated the IIP time based on the IIPCP numbers from the RMF workload activity reports.
    - The processor resource demand is 131.5 - 163.7% of a single CPU for our measurements. If the Db2ZAI instance runs on an LPAR on which other critical applications run, you should run the SA training during the lowest period of activity to avoid affecting other applications. SA training processes do not need to be run frequently. Typically, you need to rerun SA training processes only after your target Db2 subsystems go through changes that affect the baseline metrics and you change the work period definitions of the target Db2 subsystems.
- Consider putting the Spark address spaces (ALNSP\*) into a separate WLM service class with lower importance than other critical applications.
- The real storage usage requirement for the training processes is mainly above-the-bar (ATB) and for the Spark workers. From the RMF paging activity report, the maximum LPAR real storage usage increases by approximately 8 GB during the SA training process compared to when either Db2ZAI or MLZ services were not brought up on the LPAR.

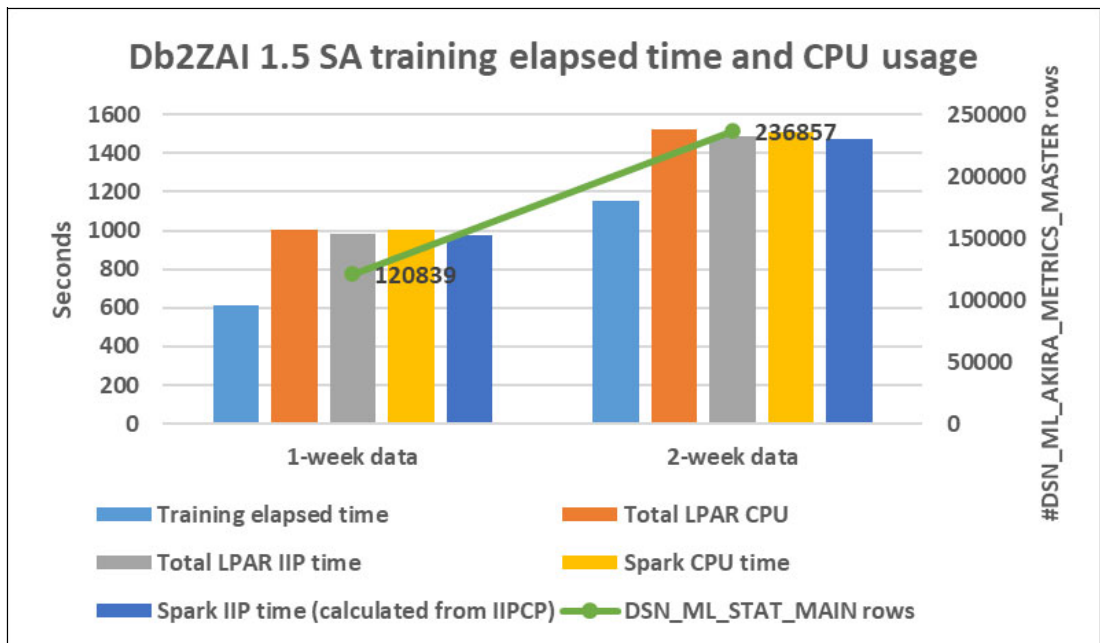


Figure 13-32 Db2ZAI 1.5 SA training elapsed time and CPU usage on the Db2ZAI UI side

## System assessments

The most typical use case for running SA is to schedule daily assessments. Every day, the scheduled assessment goes through the last 24 hours of system statistics to identify exceptions. However, administrators also can invoke on-demand SAs to assess any amount of system data. Db2ZAI 1.5 assesses Db2 subsystem statistics, local and global buffer pool statistics, and aggregated accounting statistics.

For the Db2ZAI 1.5 SA capacity evaluation, we measured the performance of three different assessments processing 24, 48, and 72 hours of data.

The base measurement is running our mixed workload on the target Db2 systems with ML started, but with no UI operations performed on the Db2ZAI UI.

The data volume that was processed for the three assessments, as recorded in `assessmentMessages.log` and the elapsed time of the three assessments, is shown in Table 13-5.

Table 13-5 Db2ZAI 1.5 system assessment study: Assessment data volume and elapsed time

Data volume	1-day count	2-day count	3-day count
StampMaster	17277	34557	51837
dfStatsBP	622008	1244088	1866168
dfStatsAcctg	69112	138232	207352
dfStatsGroupBP	311004	622044	933084
dfWorkPeriods	24	24	24
Assessment elapsed time (seconds)	3124	4007	4920

By studying the three Db2ZAI SA measurements, we made the following observations:

- The elapsed time for the assessments scales well as the data volume increases, as shown in the last row of Table 13-5.

- ▶ On the target Db2 side:
  - The statistics data in the Db2ZAI tables is retrieved through DRDA requests under the connection’s scheduler ID from the Db2 member to which Db2ZAI connects. The assessment results are written back to the Db2ZAI SA exception tables with DRDA requests.
  - As shown in Figure 13-33, the CPU cost of SAs on the Db2 target side scales well as data volume increases.
  - The general CPU overhead from the DRDA requests is below 1.5% of a processor, and the zIIP processor overhead is below 2.2% of a processor.
  - There is no real storage usage increase on the LPAR where the target Db2 subsystems run during the SA tests.

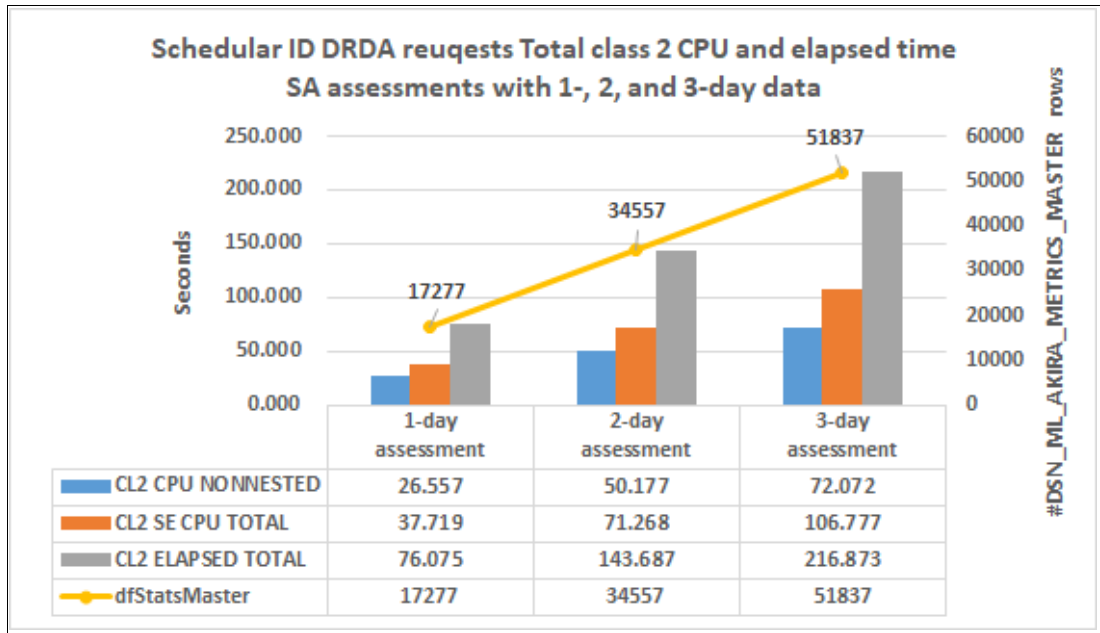


Figure 13-33 Db2ZAI 1.5 system assessment elapsed time and CPU usage on target Db2

- ▶ On the Db2ZAI UI side:
  - Db2ZAI Liberty server is the main bearer of the Db2ZAI 1.5 SA cost, and the CPU cost is 99% zIIP eligible (the IIP time is calculated from the IIPCP data in the RMF workload activity reports). As shown in Figure 13-34 on page 401, the CPU usage scales well as the data volume increases.
  - The three SAs ran with 240.1%, 261.2%, and 269.8% of a single CPU’s capacity. These measurements indicate that as many as 2.7 extra zIIP processors are needed for these SAs. If the LPAR that is hosting the Db2ZAI instance has other critical applications, you should avoid running SAs when the LPAR is busy. Also, consider putting the Db2ZAI Liberty address space into a WLM service class with lower importance than the other critical applications.
  - Unlike SA training, SAs themselves do not consume much extra real storage. The real storage usage during our assessments was no greater than the base measurement. All measurements use approximately 23.1 GB LPAR of real storage at a maximum.



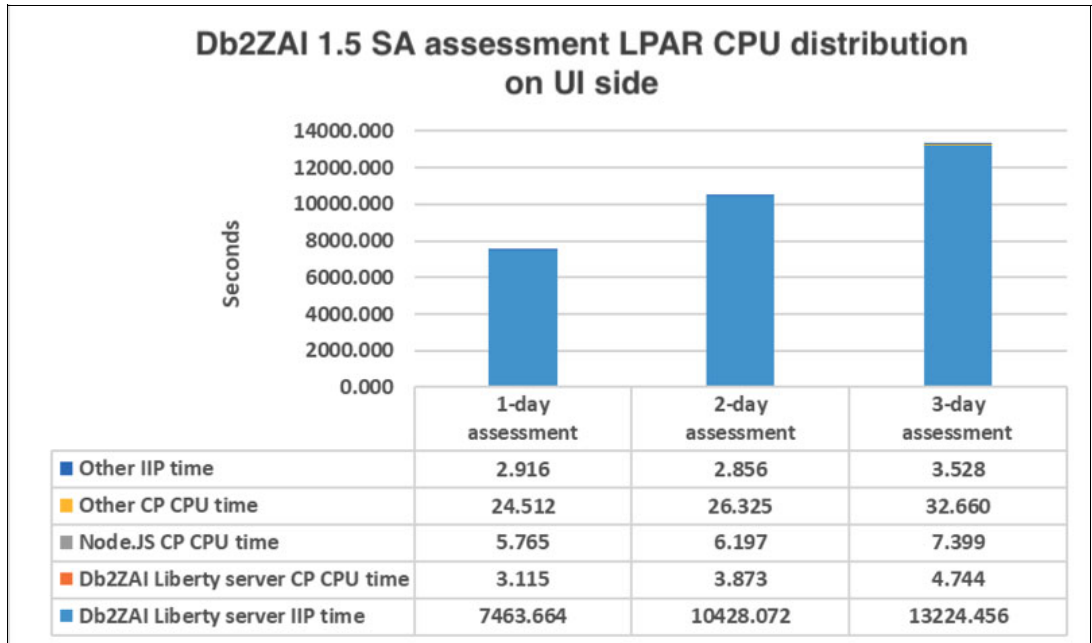


Figure 13-34 Db2ZAI 1.5 system assessment: CPU usage on the Db2ZAI UI side

### Summary of results

The following key findings summarize the results for the series of capacity evaluation tests that we did for Db2ZAI 1.5 and Db2 12 for z/OS:

- ▶ On the target Db2 subsystem side, running Db2 workloads with Db2ZAI (ML started) is inexpensive. After considering the CPU savings that the SQL optimization function can potentially bring to your workloads, the overhead will be further reduced, if not canceled entirely. Also, with proper WLM service class and report class definitions in place, running with ML started does not affect your existing applications.
- ▶ On the Db2ZAI UI side where the Db2ZAI services and WMLz run, the major CPU overhead is from the SA training and SAs. However, because the CPU overhead is close to 100% zIIP eligible, the cost is containable. If you have other zIIP dependent workloads on the same LPAR, with proper scheduling and proper WLM service classes that are defined, their performance is not affected.

## 13.6 Monitoring information

The Db2 **-DISPLAY ML** command displays the status of Db2ZAI on the target Db2 subsystems. You can issue this command against any of the target Db2 subsystems, and message DSNX611I shows the current statuses of all the Db2ZAI subtasks.

*Example 13-1 The -DISPLAY ML command output*

```

-DB2ADIS ML
DSNX611I  -DB2A DSNXODPM DISPLAY ML REPORT FOLLOWS: 840
          STATUS = STARTED
          - ML Daemon : MLZD > RUNNING
          - Db2ZAI Table Cleanup : MLTBCL > RUNNING
          SQL OPTIMIZATION = STARTED
          - Execution History Pushout : MLQEHP > RUNNING
          - AccessPath History Pushout : MLQHP > RUNNING
  
```

```
    - HV/OFNR Simple Model Training : MLSIM > RUNNING
    - Package Automation             : MLPACK > RUNNING
    - Performance Data Processing    : MLPERF > RUNNING
    - Dynamic Sql Monitoring         : MLDSMN > RUNNING
SYSTEM ASSESSMENT &
DISTRIBUTED CONNECTION CONTROL = STARTED
    - ML Generating Metrics          : MLMETX > RUNNING
DISPLAY ML REPORT COMPLETE.
DSN9022I -DB2A DSNXODPM 'DISPLAY ML' NORMAL COMPLETION
```

---

You also can obtain similar information from the Db2ZAI UI if you are using Db2ZAI 1.5 by following the instructions in [Checking the status of Db2ZAI processes on your Db2 subsystems](#).

You can monitor the statuses of the Db2ZAI services from the UI by using the instructions in [Checking the status of the Db2ZAI services](#).

For more information and complete instructions for installing, administering, and using Db2ZAI 1.5, see [Db2 AI for z/OS 1.5.0](#).



## IBM Db2 workloads

This appendix provides an overview of the workloads that are used to measure Db2 performance.

Traditionally, performance workloads started as heavy-duty batch jobs, which stressed the CPU usage and I/O of the system. For example, the original non-distributed OLTP IBM Relational Warehouse Workload (IRWW) (referred to as the classic IRWW in this publication), which was created at approximately the same time as Db2 4, is based on a retail environment that uses IMS and later CICS as the transaction monitor. In later years, more workloads that were related to distributed environments, complex OLTP, query processing, and data warehousing were added.

The following list summarizes the different types of workloads that were used to evaluate the performance of the features in this book. Not all workloads were used with all features.

- ▶ A set of the classic IRWW workloads represents the simple, legacy type, OLTP workloads that are typically run through IMS or CICS as the transaction server.
- ▶ IRWW distributed workloads are the same Db2 workloads as the classic IRWW workloads but run through remote clients, such as JDBC or remote stored procedure calls.
- ▶ The brokerage OLTP workload represents a complex OLTP workload with a mixture of lookup, update, insert, and reporting queries.
- ▶ The SAP day-posting workload is an OLTP workload with relatively simple SQL statements that are run along with intensive insert, update, and delete statements on a SAP banking database.
- ▶ The SAP account settlements workload is a batch workload that runs simple to complex SQL statements along with intensive insert, update, and delete statements against an SAP banking database.
- ▶ High-insert workloads are concurrent inserts that are run through more than 100 JCC T4 connections in data sharing.
- ▶ Special high-insert batch workloads (HIWs) are designed to compare physical design options and different structures to minimize contention.
- ▶ TSO batches represent a batch workload that is running various batch operations in a loop, including **SELECT**, **FETCH**, **UPDATE**, **INSERT**, and **DELETE**.

- ▶ A set of query workloads is run serially, mostly by using CPU parallelism. Each query workload consists of a set of 20 - 100 queries per workload.
- ▶ The IBM Cognos® BI-day short workload is a set of short running reporting queries that are concurrently run.
- ▶ The utilities scenarios are a set of various utility job executions.
- ▶ XML scenarios also are a set of various XML workloads, including OLTP, queries, batch, and utilities against XML data types.

This appendix includes the following topics:

- ▶ IBM Relational Warehouse Workload
- ▶ Relational transaction workload
- ▶ IRWW distributed workload
- ▶ IBM brokerage transaction workload
- ▶ Sample performance regression workload

## IBM Relational Warehouse Workload

The classic IRWW workload is an OLTP workload that consists of seven transactions. Each transaction consists of one or more SQL statements, each performing a distinct business function in a predefined mix.

The transactions that are used in the classic IRWW workload are listed here:

<b>Delivery</b>	Performs various <b>SELECT</b> , <b>UPDATE</b> , and <b>DELETE</b> transactions that support the delivery of a group of orders and runs as 2% of the total transaction mix.
<b>New Order</b>	Performs various <b>SELECT</b> , <b>FETCH</b> , <b>UPDATE</b> , and <b>INSERT</b> transactions that support the receipt of new customer orders and runs as 22% of the total transaction mix.
<b>Order Status</b>	Performs various <b>SELECT</b> and <b>FETCH</b> transactions that support providing the order status and runs as 25% of the total transaction mix.
<b>Payment</b>	Performs <b>SELECT</b> , <b>FETCH</b> , <b>UPDATE</b> , and <b>INSERT</b> transactions that support received customer payments and runs at 21% of the total transaction mix.
<b>Price Change</b>	Performs an <b>UPDATE</b> that supports changing the price of an item and runs as 1% of the total transaction mix.
<b>Price Quote</b>	Performs various <b>SELECT</b> transactions that support providing the price of a set of items and runs as 25% of the total transaction mix.
<b>Stock Level</b>	Performs a <b>JOIN</b> and various <b>SELECT</b> transactions that support providing the current stock level of an item and runs at 4% of the mix.

The IRWW database contains several inventory stock warehouses and sales districts. The front-end transaction manager is typically IMS, but CICS is used sometimes.

Historically, the classic IRWW workload used IMS Fast Path (IFP) regions for transaction runs. IFPs bypass IMS queuing, which allows more efficient processing, including thread reuse.

However, the new IMS connection pooling support is not applicable to IMS IFP regions. Therefore, the classic IRWW workload setup was changed to use IMS message processing regions (MPRs) to schedule and run the programs that correspond to different transactions.

**Note:** Because of this change, previously published IRWW results (for example, from Db2 10 for z/OS benchmarks) must not be compared to IRWW runs that use Db2 13 for z/OS.

The processing limit count (**PLCT**) parameter is used to control the maximum number of messages that are sent to the application program by the IMS control program for processing before the application program is reloaded in the message processing program (MPP) region. This value is set to 65535 for two-thirds of the transactions and to 0 for the remaining transactions.

**PLCT(0)** means the maximum number of messages that are sent to the application is one, and the application program is reloaded into the MPP region before receiving a subsequent message. **PLCT(65535)** means that no limit is placed upon the number of messages that are processed at single program load.

## Relational transaction workload

The RTW is a standard workload that is used by the CICS performance team to assess changes in performance characteristics for new CICS releases for applications that access Db2. The Db2 development team decided to use the RTW as one of its performance test benchmark workloads to get better performance analysis coverage for CICS and Db2 type of workloads. The workload was modified slightly compared to the version that the CICS team uses to better reflect the Db2 behavior of the workload.

The RTW features the following characteristics:

- ▶ All programs that make up the workload are written in COBOL.
- ▶ The workload uses 20 (data) tables, 18 of which are in partition-by-growth (PBG) table spaces, and two of which are in segmented table spaces (many Db2 clients still have legacy segmented table spaces, so regression coverage is still needed for them). The workload uses 11 non-unique indexes and 16 unique indexes.
- ▶ RTW issues on average 200 Db2 calls per transaction.
- ▶ The transaction mix consists of 26% select, 3% insert, 7% update, 3% delete, 6% open cursor, 36% fetch cursor, and 6% close cursor operations.

The following environment configuration is used for the non-data-sharing RTW workload:

- ▶ One z15 logical partition (LPAR) with eight general-purpose processors (CPs) and two IBM zSystems Integrated Information Processors (zIIPs)
- ▶ Four CICS regions running CICS Transaction Server (CICS TS) 5.6
- ▶ A non-data-sharing Db2 subsystem
- ▶ z/OS 2.5
- ▶ An IBM DS8870 direct access storage device (DASD) controller
- ▶ Teleprocessing Network Simulator (TPNS) on another z15 LPAR with four general CPs to drive the workload

The following environment is used by the two-way data-sharing RTW:

- ▶ Two z15 LPARs with eight general CPs and two zIIPs each
- ▶ Eight CICS regions running CICS TS 5.6 with four regions running on each of the LPARs
- ▶ z/OS 2.5
- ▶ A DS8870 DASD controller
- ▶ TPNS on another z15 LPAR with four general CPs to drive the workload
- ▶ Two z15 Internal Coupling Facility (ICF) LPARs running CFLEVEL 25 CFCC, and each connected to the z/OS LPARs through two CS5 channels

## IRWW distributed workload

The IRWW distributed workload was created by converting the classic IRWW workload to run in distributed environments. It consists of seven transactions that support a product warehouse, customer order, and order delivery system that runs in a distributed environment. Each transaction consists of approximately 25 Data Manipulation Language (DML) operations on average, with seven of those DML operations being **INSERT**, **UPDATE**, or **DELETE**. The IRWW distributed workloads consist of stored procedure and non-stored procedure workloads.

For more information about the seven transaction types, a brief description of each, and the percentage of the transaction mix, see “IBM Relational Warehouse Workload” on page 404.

## IBM brokerage transaction workload

The IBM brokerage transaction workload is a more complex OLTP workload than the other OLTP workloads. It runs various SQL procedures to simulate transactions in a brokerage firm by using a mixture of lookup, update, insert, and reporting queries.

This OLTP workload accesses 33 tables that contain approximately 15,000 Db2 database objects. The brokerage workload represents transactions of customers in the financial market who are updating account information. The transactions are implemented in stored procedure by using native SQL stored procedures. The workload uses the following resources:

- ▶ 1.3 TB of data
- ▶ Thirteen buffer pools
- ▶ Thirty-three table spaces
- ▶ Sixty-eight indexes

Two versions of IBM brokerage transaction workload are used. Version 1 excludes the brokerage and data maintenance transactions from the workload. Version 2 contains the full set of transactions.

## Sample performance regression workload

IBM provided an enhanced a sample performance regression workload to work with Db2 11 and later. This workload is available for customers to use for basic performance regression in their environments. For more information and detailed instructions for installing this workload onto your LPAR, customizing the scripts and JCL for your environment, and building the database to run the sample workload, see [GitHub](#).







# Artificial intelligence semantic queries

This appendix shows the artificial intelligence (AI) semantic SQL queries that were used to evaluate the performance of the Db2 13 SQL Data Insights feature. For more information, see Chapter 6, “SQL Data Insights” on page 211.

This appendix includes the following topics:

- ▶ SQL Data Insights AI semantic queries that are used in scenario 1 and scenario 2
- ▶ SQL Data Insights semantic queries that are used in scenario 3
- ▶ SQL Data Insights semantic queries that are used in scenario 4

# SQL Data Insights AI semantic queries that are used in scenario 1 and scenario 2

The AI object that is used by the queries in these scenarios is a table that has eight columns: Seven of the columns are trained as categorical data type columns, and one column is trained as a numerical data type column. The cardinality of the AI object is approximately 1.3 million rows, and the model table is approximately 80,000 rows.

## AI semantic queries that use the AI\_SIMILARITY function

The following AI semantic queries use the AI\_SIMILARITY built-in scalar function in scenarios 1 and 2.

### T00SIMX

```
SELECT DISTINCT VENDOR_NAME, SIMILY
FROM
(
SELECT
  VENDOR_NAME,
  AI_SIMILARITY (
    VENDOR_NAME, 'VERIZON' USING MODEL COLUMN VENDOR_NAME)
  AS SIMILY
FROM USRT031.VIRG1TB
)
WHERE SIMILY IS NOT NULL
AND TRIM(VENDOR_NAME) <> 'VERIZON'
ORDER BY SIMILY DESC
FETCH FIRST 10 ROWS ONLY
WITH UR;
```

### T01SIMX

```
SELECT
  VENDOR_NAME,
  SIMILY,
  SUM(AMOUNT) AS TOTAL_AMOUNT,
  AVG(AMOUNT) AS AVG_AMOUNT,
  MIN(AMOUNT) AS MIN_AMOUNT,
  MAX(AMOUNT) AS MAX_AMOUNT,
  COUNT(*) AS COUNT_TXNS
FROM
(
SELECT
  VENDOR_NAME,
  AMOUNT,
  AI_SIMILARITY(
    VENDOR_NAME, 'VERIZON' USING MODEL COLUMN VENDOR_NAME
  ) AS SIMILY
FROM USRT031.VIRG1TB
WHERE AI_SIMILARITY(
  VENDOR_NAME, 'VERIZON' USING MODEL COLUMN VENDOR_NAME
) IS NOT NULL
) TMP
```

```

GROUP BY
  VENDOR_NAME,
  SIMILY
ORDER BY SIMILY DESC
FETCH FIRST 6 ROWS ONLY
WITH UR;

```

### **T02SIMX**

```

SELECT
  DISTINCT
    EXP.AGY_AGENCY_KEY,
    AGY.AGY_AGENCY_NAME,
    AI_SIMILARITY(
      EXP.AGY_AGENCY_KEY, 125 USING MODEL COLUMN EXP.AGY_AGENCY_KEY
    ) AS SIMILY
FROM USRT031.VIRG1TB EXP
INNER JOIN USRT031.VIRGAGY AGY
  ON EXP.AGY_AGENCY_KEY = AGY.AGY_AGENCY_KEY
WHERE
  AI_SIMILARITY(
    EXP.AGY_AGENCY_KEY, 125 USING MODEL COLUMN EXP.AGY_AGENCY_KEY
  ) IS NOT NULL
  AND EXP.AGY_AGENCY_KEY <> 125
ORDER BY 3 DESC
FETCH FIRST 10 ROWS ONLY
WITH UR;

```

### **T03SIMX**

```

SELECT
  YEAR(VOUCHER_DATE) AS YR,
  MONTH(VOUCHER_DATE) AS MTH,
  SIMILAR.AGY_AGENCY_KEY,
  SIMILAR.AGY_AGENCY_NAME,
  SUM(AMOUNT) AS TOTAL_AMOUNT,
  RANK() OVER (PARTITION BY
    YEAR(VOUCHER_DATE),
    MONTH(VOUCHER_DATE)
    ORDER BY SUM(AMOUNT) DESC
  ) AS RANKING
FROM
  USRT031.VIRG1TB EX,
  (
  SELECT
    DISTINCT
      EXP.AGY_AGENCY_KEY,
      AGY.AGY_AGENCY_NAME,
      AI_SIMILARITY(
        EXP.AGY_AGENCY_KEY, 125 USING MODEL COLUMN EXP.AGY_AGENCY_KEY
      ) AS SIMILY
    FROM USRT031.VIRG1TB EXP
    INNER JOIN USRT031.VIRGAGY AGY
      ON EXP.AGY_AGENCY_KEY = AGY.AGY_AGENCY_KEY
  WHERE
    AI_SIMILARITY(

```

```

        EXP.AGY_AGENCY_KEY, 125 USING MODEL COLUMN EXP.AGY_AGENCY_KEY
    ) IS NOT NULL
ORDER BY 3 DESC
FETCH FIRST 10 ROWS ONLY
) SIMILAR
WHERE EX.AGY_AGENCY_KEY = SIMILAR.AGY_AGENCY_KEY
GROUP BY
    YEAR(VOUCHER_DATE),
    MONTH(VOUCHER_DATE),
    SIMILAR.AGY_AGENCY_KEY,
    SIMILAR.AGY_AGENCY_NAME
ORDER BY 1, 2, 6
WITH UR;

```

### **T04SIMX**

```

SELECT * FROM
(
SELECT DISTINCT
    'DISSIMILAR' AS SIMILARITY_TYPE,
    EXP.AGY_AGENCY_KEY,
    AGY.AGY_AGENCY_NAME,
    AI_SIMILARITY(
        EXP.AGY_AGENCY_KEY USING MODEL COLUMN EXP.AGY_AGENCY_KEY,
        'COUNTY WASTE LLC' USING MODEL COLUMN EXP.VENDOR_NAME) AS SIMILY
FROM USRT031.VIRG1TB EXP,
    USRT031.VIRGAGY AGY
WHERE
    EXP.AGY_AGENCY_KEY = AGY.AGY_AGENCY_KEY AND
    AI_SIMILARITY(
        EXP.AGY_AGENCY_KEY USING MODEL COLUMN EXP.AGY_AGENCY_KEY,
        'COUNTY WASTE LLC' USING MODEL COLUMN EXP.VENDOR_NAME)
        IS NOT NULL
ORDER BY 4
FETCH FIRST 10 ROWS ONLY
) DISSIMILR
UNION ALL
SELECT * FROM
(
SELECT DISTINCT
    'SIMILAR' AS SIMILARITY_TYPE,
    EXP.AGY_AGENCY_KEY,
    AGY.AGY_AGENCY_NAME,
    AI_SIMILARITY(
        EXP.AGY_AGENCY_KEY USING MODEL COLUMN EXP.AGY_AGENCY_KEY,
        'COUNTY WASTE LLC' USING MODEL COLUMN EXP.VENDOR_NAME) AS SIMILY
FROM USRT031.VIRG1TB EXP,
    USRT031.VIRGAGY AGY
WHERE
    EXP.AGY_AGENCY_KEY = AGY.AGY_AGENCY_KEY AND
    AI_SIMILARITY(
        EXP.AGY_AGENCY_KEY USING MODEL COLUMN EXP.AGY_AGENCY_KEY,
        'COUNTY WASTE LLC' USING MODEL COLUMN EXP.VENDOR_NAME)
        IS NOT NULL
ORDER BY 4 DESC
FETCH FIRST 10 ROWS ONLY

```

```
) SIMILR  
WITH UR;
```

## AI semantic queries that use the AI\_SEMANTIC\_CLUSTER function

The following AI semantic queries use the AI\_SEMANTIC\_CLUSTER built-in scalar function in scenarios 1 and 2.

### T00SEM1

```
SELECT  
  EXP.VENDOR_NAME,  
  SUM(EXP.AMOUNT) AS NET_EXPENDITURE,  
  VLIST.CLUSTER_PROXIMITY  
FROM  
  USRT031.VIRG1TB EXP,  
  (  
  SELECT DISTINCT VENDOR_NAME,  
    AI_SEMANTIC_CLUSTER(VENDOR_NAME,  
      'COUNTY OF FAIRFAX'  
    ) AS CLUSTER_PROXIMITY  
  FROM  
    USRT031.VIRG1TB EXP  
  WHERE  
    AI_SEMANTIC_CLUSTER(VENDOR_NAME,  
      'COUNTY OF FAIRFAX'  
    ) IS NOT NULL  
    AND VENDOR_NAME NOT IN (  
      'COUNTY OF FAIRFAX')  
  ORDER BY 2 DESC  
  FETCH FIRST 10 ROWS ONLY  
  ) VLIST  
WHERE EXP.VENDOR_NAME = VLIST.VENDOR_NAME  
GROUP BY EXP.VENDOR_NAME, VLIST.CLUSTER_PROXIMITY  
ORDER BY VLIST.CLUSTER_PROXIMITY DESC  
WITH UR;
```

### T00SEM2

```
SELECT  
  EXP.VENDOR_NAME,  
  SUM(EXP.AMOUNT) AS NET_EXPENDITURE,  
  VLIST.CLUSTER_PROXIMITY  
FROM  
  USRT031.VIRG1TB EXP,  
  (  
  SELECT DISTINCT VENDOR_NAME,  
    AI_SEMANTIC_CLUSTER(VENDOR_NAME,  
      'COUNTY OF HENRICO',  
      'COUNTY OF FAIRFAX'  
    ) AS CLUSTER_PROXIMITY  
  FROM  
    USRT031.VIRG1TB EXP  
  WHERE  
    AI_SEMANTIC_CLUSTER(VENDOR_NAME,  
      'COUNTY OF HENRICO',
```

```

        'COUNTY OF FAIRFAX'
    ) IS NOT NULL
    AND VENDOR_NAME NOT IN (
        'COUNTY OF HENRICO',
        'COUNTY OF FAIRFAX')
ORDER BY 2 DESC
FETCH FIRST 10 ROWS ONLY
) VLIST
WHERE EXP.VENDOR_NAME = VLIST.VENDOR_NAME
GROUP BY EXP.VENDOR_NAME, VLIST.CLUSTER_PROXIMITY
ORDER BY VLIST.CLUSTER_PROXIMITY DESC
WITH UR;

```

### **T00SEM3**

```

SELECT
    EXP.VENDOR_NAME,
    SUM(EXP.AMOUNT) AS NET_EXPENDITURE,
    VLIST.CLUSTER_PROXIMITY
FROM
    USRT031.VIRG1TB EXP,
    (
        SELECT DISTINCT VENDOR_NAME,
            AI_SEMANTIC_CLUSTER(VENDOR_NAME,
                'COUNTY OF LOUDOUN',
                'COUNTY OF HENRICO',
                'COUNTY OF FAIRFAX'
            ) AS CLUSTER_PROXIMITY
        FROM
            USRT031.VIRG1TB EXP
        WHERE
            AI_SEMANTIC_CLUSTER(VENDOR_NAME,
                'COUNTY OF LOUDOUN',
                'COUNTY OF HENRICO',
                'COUNTY OF FAIRFAX'
            ) IS NOT NULL
            AND VENDOR_NAME NOT IN (
                'COUNTY OF LOUDOUN',
                'COUNTY OF HENRICO',
                'COUNTY OF FAIRFAX')
        ORDER BY 2 DESC
        FETCH FIRST 10 ROWS ONLY
    ) VLIST
WHERE EXP.VENDOR_NAME = VLIST.VENDOR_NAME
GROUP BY EXP.VENDOR_NAME, VLIST.CLUSTER_PROXIMITY
ORDER BY VLIST.CLUSTER_PROXIMITY DESC
WITH UR;

```

## AI semantic queries that use the AI\_ANALOGY function

The following AI semantic queries use the **AI\_ANALOGY** built-in scalar function in scenarios 1 and 2.

### T00ANAX

```
SELECT
  DISTINCT OBJ_OBJECT_NAME, VENDOR_NAME,
  AI_ANALOGY(
    '132'                               USING MODEL COLUMN EXP.OBJ_OBJECT_KEY,
    'TREASURER OF VA MEDICAID ACCTS' USING MODEL COLUMN VENDOR_NAME,
    '343'                               USING MODEL COLUMN EXP.OBJ_OBJECT_KEY,
    VENDOR_NAME) AS ANALOGY_SCORE
FROM USRTO31.VIRG1TB EXP, USRTO31.VIRGOBJ OBJ
WHERE
  EXP.OBJ_OBJECT_KEY = OBJ.OBJ_OBJECT_KEY
  AND AI_ANALOGY(
    '132'                               USING MODEL COLUMN EXP.OBJ_OBJECT_KEY,
    'TREASURER OF VA MEDICAID ACCTS' USING MODEL COLUMN VENDOR_NAME,
    '343'                               USING MODEL COLUMN EXP.OBJ_OBJECT_KEY,
    VENDOR_NAME) IS NOT NULL
ORDER BY 3 DESC
FETCH FIRST 10 ROWS ONLY
WITH UR;
```

### T01ANAX

```
SELECT DISTINCT VENDOR_NAME, SIMILARITY
FROM
  (SELECT VENDOR_NAME, AI_ANALOGY(
    '7006'                               USING MODEL COLUMN FNDDTL_FUND_DETAIL_KEY,
    'COUNTY OF ARLINGTON' USING MODEL COLUMN VENDOR_NAME,
    '514'                               USING MODEL COLUMN FNDDTL_FUND_DETAIL_KEY,
    VENDOR_NAME) AS SIMILARITY
  FROM USRTO31.VIRG1TB
  )
WHERE SIMILARITY > 0.0
ORDER BY SIMILARITY DESC
FETCH FIRST 10 ROWS ONLY
WITH UR;
```

## SQL Data Insights semantic queries that are used in scenario 3

The AI object that is used by the queries in this scenario is a table that has eight columns: Seven of the columns are trained as categorical data type columns, and one column is trained as a numerical data type column. The cardinality of the AI object is approximately 1.3 million rows, and the model table is approximately 80,000 rows.

### AI semantic query that uses the AI\_SIMILARITY function

The following AI semantic query uses the AI\_SIMILARITY built-in scalar function in scenario 3.

#### T00SIMX

```
SELECT DISTINCT VENDOR_NAME, SIMILY
FROM
(
SELECT
  VENDOR_NAME,
  AI_SIMILARITY (
    VENDOR_NAME, 'VERIZON' USING MODEL COLUMN VENDOR_NAME)
  AS SIMILY
FROM USRT031.VIRG1TB
)
WHERE SIMILY IS NOT NULL
AND TRIM(VENDOR_NAME) <> 'VERIZON'
ORDER BY SIMILY DESC
FETCH FIRST 10 ROWS ONLY
WITH UR;
```

### AI semantic query that uses the AI\_SEMANTIC\_CLUSTER function

The following AI semantic query uses the AI\_SEMANTIC\_CLUSTER built-in scalar function was used in scenario 3.

#### T00SEM3

```
SELECT
  EXP.VENDOR_NAME,
  SUM(EXP.AMOUNT) AS NET_EXPENDITURE,
  VLIST.CLUSTER_PROXIMITY
FROM
  USRT031.VIRG1TB EXP,
(
  SELECT DISTINCT VENDOR_NAME,
    AI_SEMANTIC_CLUSTER(VENDOR_NAME,
      'COUNTY OF LOUDOUN',
      'COUNTY OF HENRICO',
      'COUNTY OF FAIRFAX'
    ) AS CLUSTER_PROXIMITY
) AS CLUSTER_PROXIMITY
FROM
  USRT031.VIRG1TB EXP
WHERE
  AI_SEMANTIC_CLUSTER(VENDOR_NAME,
    'COUNTY OF LOUDOUN',
```



```

        'COUNTY OF HENRICO',
        'COUNTY OF FAIRFAX'
    ) IS NOT NULL
    AND VENDOR_NAME NOT IN (
        'COUNTY OF LOUDOUN',
        'COUNTY OF HENRICO',
        'COUNTY OF FAIRFAX')
    ORDER BY 2 DESC
    FETCH FIRST 10 ROWS ONLY
) VLIST
WHERE EXP.VENDOR_NAME = VLIST.VENDOR_NAME
GROUP BY EXP.VENDOR_NAME, VLIST.CLUSTER_PROXIMITY
ORDER BY VLIST.CLUSTER_PROXIMITY DESC
WITH UR;

```

## AI semantic query that uses the AI\_ANALOGY function

The following AI semantic query uses the **AI\_ANALOGY** built-in scalar function in scenario 3.

### T01ANAX

```

SELECT DISTINCT VENDOR_NAME, SIMILARITY
FROM
  (SELECT VENDOR_NAME, AI_ANALOGY(
    '7006'          USING MODEL COLUMN FNDDTL_FUND_DETAIL_KEY,
    'COUNTY OF ARLINGTON' USING MODEL COLUMN VENDOR_NAME,
    '514'          USING MODEL COLUMN FNDDTL_FUND_DETAIL_KEY,
    VENDOR_NAME) AS SIMILARITY
  FROM USRTO31.VIRG1TB
  )
WHERE SIMILARITY > 0.0
ORDER BY SIMILARITY DESC
FETCH FIRST 10 ROWS ONLY
WITH UR;

```

## SQL Data Insights semantic queries that are used in scenario 4

All the AI objects that are used by the queries in this scenario have 29 columns: Fifteen columns are trained as categorical data type columns, and 14 columns are trained as numerical data type columns.

Table B-1 shows the views that are used by each of the SQL DI function types.

*Table B-1 Views that are used by each SQL DI function type*

View name	Cardinality of the view (rows)	Cardinality of the view's base table (rows)	Cardinality of the associated model table (rows)
VIEW1	500,000	45 million	272,000
VIEW2	2 million	45 million	679,000
VIEW3	4 million	45 million	1.7 million
VIEW4	10 million	45 million	4 million

View name	Cardinality of the view (rows)	Cardinality of the view's base table (rows)	Cardinality of the associated model table (rows)
VIEW5	45 million	45 million	4.4 million
VIEW6	101 million	101 million	8.1 million

## AI semantic query that uses the AI\_SIMILARITY function

The following AI semantic query uses the **AI\_SIMILARITY** built-in scalar function in scenario 4.

### FOOSIMX

```

SELECT DISTINCT LOAN_SEQ_NUM, SIMILY,
               CREDIT_SCORE, CURRENT_LOAN_DLQCY_STATUS, LOAN_PURPOSE
FROM
(
SELECT
  LOAN_SEQ_NUM,
  AI_SIMILARITY(
    LOAN_SEQ_NUM, 'F19Q10000002' USING MODEL COLUMN LOAN_SEQ_NUM)
  AS SIMILY,
  CREDIT_SCORE,
  CURRENT_LOAN_DLQCY_STATUS,
  LOAN_PURPOSE
FROM FMACE1.VIEW1
)
WHERE SIMILY IS NOT NULL
AND LOAN_SEQ_NUM <> 'F19Q10000002'
ORDER BY SIMILY DESC
FETCH FIRST 10 ROWS ONLY
WITH UR;

```

## AI semantic query that uses the AI\_SEMANTIC\_CLUSTER function

The following AI semantic query uses the **AI\_SEMANTIC\_CLUSTER** built-in scalar function in scenario 4.

### FOOSEMX

```

SELECT LOAN_SEQ_NUM, SIMILARITY
FROM
  (SELECT LOAN_SEQ_NUM,
         AI_SEMANTIC_CLUSTER
         (LOAN_SEQ_NUM, 'F19Q10000097', 'F19Q10000098', 'F19Q10000099')
         AS SIMILARITY
   FROM FMACE1.VIEW1
   WHERE
     LOAN_SEQ_NUM NOT IN ('F19Q10000097', 'F19Q10000098', 'F19Q10000099')
   GROUP BY LOAN_SEQ_NUM
  )
WHERE SIMILARITY > 0
ORDER BY SIMILARITY DESC, LOAN_SEQ_NUM ASC
FETCH FIRST 5 ROWS ONLY
WITH UR;

```

## AI semantic query that uses the AI\_ANALOGY function

The following AI semantic query uses the **AI\_ANALOGY** built-in scalar function in scenario 4.

### **F00ANAX**

```
SELECT DISTINCT LOAN_SEQ_NUM,PROPERTY_TYPE,SIMILARITY
FROM
  (SELECT LOAN_SEQ_NUM,PROPERTY_TYPE, AI_ANALOGY(
    'P' USING MODEL COLUMN OCCC_STATUS,
    'SF' USING MODEL COLUMN PROPERTY_TYPE,
    'I' USING MODEL COLUMN OCCC_STATUS,
    PROPERTY_TYPE) AS SIMILARITY
  FROM FMACE1.VIEW1
 )
WHERE SIMILARITY > 0.0
ORDER BY SIMILARITY DESC
FETCH FIRST 10 ROWS ONLY
WITH UR;
```





# **IBM Db2 Analytics Accelerator for z/OS workloads**

The query workload descriptions and results in Chapter 10, “Performance enhancements for IDAA for z/OS and IBM Db2 for z/OS Data Gate” on page 307 mentioned the internal TPC-H and Customer FD workloads that were used for testing. This appendix briefly describes what these workloads consist of and why they were used.

This appendix includes the following topics:

- ▶ TPC-H query workload
- ▶ Customer FD workload

## TPC-H query workload

The Transaction Processing Performance Council (TPC) is an international association. Some of the largest IT companies are members of the TPC. As a result of their mission to provide reliable, standardized tests for transactional database operations, they created several benchmarks.

One of those benchmarks is the TPC-H decision support benchmark for database systems. It tests the ability for the system to handle large volumes of data by using complex queries. Therefore, our own test workload uses the general framework of the TPC-H framework, with some small modifications.

For more information about the TPC-H benchmark, including details about queries, tables, and so on, see the [TPC home page](#).

## Customer FD workload

The Customer FD performance workload is a comprehensive test suite with over 1000 queries. These queries are based on customer queries from numerous Db2 proof-of-concept (PoC) projects with extra sets of scripts for extract-transform-load (ETL)-like workloads. It consists of a star schema with snowflake dimension tables, one large and one small fact table, and a bitemporal table. The queries of the Customer FD performance workload are organized in groups, as shown in Table C-1.

*Table C-1 Groups of queries for the Customer FD workload*

Group name	Characteristics
G1	Plain table scans with different number of selected columns and no join.
G2	Detailed buffer pool tests with scans.
G3	Scans with one or more joins, and tests the performance of joins.
G4	Compares filtering on dimensions with filtering on fact.
G6	Group by with no joins other than Period and CustDim.
G7	IN list.
G8	Compares filtering methods, including IN list, Snowflake, CTE, and Temp table.
G9	Foreign key, join filtering, and zonemap-aware joins.
G10	Foreign key, join filtering, and zonemap-aware joins. Specialized on MAKO.
G11	Optimizer and joins.
G12	Optimizer pushdown, group by, and union.
G13	Join fast with medium-sized tables.
G14	Compare Period.
G16	Count and Join Distinct.
G17	Group by, Having, and count distinct.
G18	Expressions, Case, and functions.
G19	Statistics and OLAP window.

<b>Group name</b>	<b>Characteristics</b>
G21	Correlation.
G22	Group by with different types of joins and encodings.
G25	Customer POC query 06.





# Abbreviations and acronyms

<b>AI</b>	artificial intelligence	<b>ETR</b>	external throughput rate
<b>AOT</b>	accelerator-only table	<b>FL</b>	function level
<b>APN</b>	absolute page numbering	<b>FLR</b>	fixed-length record
<b>AREOR</b>	advisory reorg	<b>FPGA</b>	Field Programmable Gate Array
<b>ATB</b>	above-the-bar	<b>FTB</b>	fast traversal blocks
<b>BiF</b>	built-in function	<b>GA</b>	general availability
<b>BP0</b>	Buffer pool 0	<b>GBP</b>	group buffer pool
<b>BSDS</b>	bootstrap data set	<b>GPC</b>	general-purpose central processor
<b>BTB</b>	below-the-bar	<b>HA</b>	highly available
<b>CDC</b>	Change Data Capture	<b>HIW</b>	high-insert batch workload
<b>CF</b>	Coupling Facility	<b>HMC</b>	Hardware Management Console
<b>CI</b>	Control Interval	<b>HWM</b>	high-water mark
<b>CIB</b>	coupling over InfiniBand	<b>IAS</b>	IBM Integrated Analytics System
<b>CICS TS</b>	CICS Transaction Server	<b>IBM</b>	International Business Machines Corporation
<b>CL5</b>	Long Reach Coupling	<b>ICF</b>	Internal Coupling Facility
<b>CM</b>	conversion mode	<b>ICP</b>	Internal Coupling
<b>CP</b>	central processor	<b>IDAA</b>	IBM Db2 Analytics Accelerator
<b>CPACF</b>	CP Assist for Cryptographic Functions	<b>IFC</b>	Instrumentation Facility Component
<b>CPC</b>	central processor complex	<b>IFCID</b>	Instrumentation Facility Component Identifier
<b>DASD</b>	direct access storage device	<b>IFI</b>	Instrumentation Facility Interface
<b>Db2ZAI</b>	IBM Db2 AI for z/OS	<b>IFL</b>	Integrated Facility on Linux
<b>DBAT</b>	database access thread	<b>IFP</b>	IMS Fast Path
<b>DBD</b>	database descriptor	<b>IOS</b>	input/output supervisor
<b>DCC</b>	Distributed Connection Control	<b>IPL</b>	initial program load
<b>DCRM</b>	Data Communications Resource Manager	<b>IRLM</b>	internal resource lock manager
<b>DDL</b>	data definition statements	<b>IRWW</b>	IBM Relational Warehouse Workload
<b>DML</b>	Data Manipulation Language	<b>ISMF</b>	Integrated Storage Management Facility
<b>DRDA</b>	Distributed Relational Database Architecture	<b>ITR</b>	internal throughput rate
<b>DSTATSRT</b>	distribution statistics sort	<b>LOB</b>	large object
<b>DWDM</b>	dense wavelength division multiplexing	<b>LPAR</b>	logical partition
<b>EBCDIC</b>	Extended Binary Coded Decimal Interchange Code	<b>ML</b>	machine learning
<b>ECSA</b>	extended common service area	<b>MLN</b>	multiple logical nodes
<b>EDM</b>	environmental descriptor manager	<b>MPP</b>	message processing program
<b>ENFM</b>	enabling new function mode	<b>MPR</b>	message processing region
<b>EPROC</b>	Expansion Preprocessor Routine	<b>MRI</b>	multiple-row insert
<b>ESQA</b>	Enhanced System Queue Area	<b>NFM</b>	new function mode
<b>ETL</b>	extract-transform-load	<b>NLI</b>	NOT LOGGED INITIALLY

<b>NPI</b>	non-partitioned index	<b>WLB</b>	workload balancing
<b>OBID</b>	object identifier	<b>WLM</b>	Workload Manager
<b>ODF</b>	Red Hat OpenShift Data Foundation	<b>WMLz</b>	IBM Watson Machine Learning for z/OS
<b>OLTP</b>	Online Transaction Processing	<b>zAIO</b>	IBM Z AI Optimization Library
<b>OMPE</b>	IBM OMEGAMON for Db2 Performance Expert	<b>zBNA</b>	IBM Z Batch Network Analyzer
<b>PBG</b>	partition-by-growth	<b>zDNN</b>	IBM Z Deep Neural Network Library
<b>PBR</b>	partition-by-range	<b>zEDC</b>	IBM zEnterprise Data Compression
<b>PFID</b>	PCIe Function Identifier	<b>zHPF</b>	IBM Z High-Performance FICON
<b>PI</b>	partitioned index	<b>zIIP</b>	IBM zSystems Integrated Information Processor
<b>PLCT</b>	processing limit count		
<b>PoC</b>	proof-of-concept		
<b>PRB</b>	performance regression bucket		
<b>PSID</b>	pageset ID		
<b>RDS</b>	Relational Data System		
<b>RLL</b>	row-level locking		
<b>RNI</b>	relative nest intensity		
<b>RPN</b>	relative page numbering		
<b>RPO</b>	recovery point objective		
<b>RSM</b>	Real Storage Manager		
<b>RTO</b>	recovery time objective		
<b>RTS</b>	real-time statistics		
<b>RTW</b>	relational transaction workload		
<b>SA</b>	system assessment		
<b>SCA</b>	shared communications area		
<b>SIMD</b>	single instruction, multiple data		
<b>SLA</b>	service level agreement		
<b>SMF</b>	System Management Facility		
<b>SORTL</b>	SORT LISTS		
<b>SQA</b>	System Queue Area		
<b>SRB</b>	System Recovery Boost		
<b>SSID</b>	subsystem ID		
<b>SWB</b>	scheduler work block		
<b>TDE</b>	Transparent Data Encryption		
<b>TLS</b>	Transport Layer Security		
<b>TPC</b>	Transaction Processing Performance Council		
<b>TPNS</b>	Teleprocessing Network Simulator		
<b>TS STATS SPHS</b>	table space statistics subphase		
<b>UDF</b>	user-defined function		
<b>UTS</b>	universal table space		
<b>VLR</b>	variable length record		
<b>WAR</b>	Write-And-Register		
<b>WARM</b>	Write-And-Register-Multiple		

# Related publications

The publications that are listed in this section are considered suitable for a more detailed description of the topics that are covered in this book.

## IBM Redbooks

The following IBM Redbooks publications provide additional information about the topics in this document. Some publications that are referenced in this list might be available in softcopy only.

- ▶ *Getting Started with IBM zHyperLink for z/OS*, REDP-5493
- ▶ *IBM Db2 13 for z/OS and More*, SG24-8527
- ▶ *IBM Db2 12 for z/OS Performance Topics*, SG24-8404
- ▶ *IBM z16 (3931) Technical Guide*, SG24-8951
- ▶ *Introducing IBM Z System Recovery Boost*, REDP-5563

You can search for, view, download, or order these documents and other Redbooks, Redpapers, web docs, drafts, and additional materials, at the following website:

[ibm.com/redbooks](https://ibm.com/redbooks)

## Online resources

These websites are also relevant as further information sources:

- ▶ Db2 13 for z/OS documentation  
<https://www.ibm.com/docs/en/db2-for-zos/13>
- ▶ *Db2 13 for z/OS - What's New?*  
[https://www.ibm.com/docs/en/SSEPEK\\_13.0.0/pdf/db2z\\_13\\_wnewbook.pdf](https://www.ibm.com/docs/en/SSEPEK_13.0.0/pdf/db2z_13_wnewbook.pdf)
- ▶ Db2 Real Storage Management and Recommendations  
<https://community.ibm.com/community/user/hybridatamanagement/blogs/paul-mcwilliams1/2022/04/12/db2-real-storage-management?CommunityKey=621c2a2a-01f9-4b57-992f-36ed7432e3bb>
- ▶ Evaluation of zHyperLink Write for Db2 Active Logging with SAP Banking on IBM Z  
<https://www.ibm.com/support/pages/evaluation-zhyperlink-write-db2-active-logging-sap-banking-z-0>
- ▶ IBM Data Management Community  
<https://community.ibm.com/community/user/hybridatamanagement/home>
- ▶ IBM Db2 AI for z/OS 1.5.0 documentation  
<https://www.ibm.com/docs/en/db2-ai-for-zos/1.5.0>

- ▶ IBM Open Data Analytics for z/OS  
<https://www.ibm.com/docs/en/izoda/1.1.0?topic=spark-configuring-memory-cpu-options>
- ▶ IBM Z Batch Network Analyzer (zBNA) Tool  
<https://www.ibm.com/support/pages/ibm-z-batch-network-analyzer-zbna-tool-0>
- ▶ IBM Z and LinuxONE Community  
<https://www.ibm.com/community/z/>
- ▶ Manuals for Db2 13 for z/OS  
<https://www.ibm.com/docs/en/db2-for-zos/13?topic=zos-pdf-format-manuals-db2-13>
- ▶ z/OS 2.5.0 documentation  
<https://www.ibm.com/docs/en/zos/2.5.0>

## Help from IBM

IBM Support and downloads

[ibm.com/support](https://www.ibm.com/support)

IBM Global Services

[ibm.com/services](https://www.ibm.com/services)



**Redbooks**

# IBM Db2 13 for z/OS Performance Topics

SG24-8536-00

ISBN 0738461008



(1.0" spine)

0.875" x 1.498"

460 <-> 788 pages







SG24-8536-00

ISBN 0738461008

Printed in U.S.A.

Get connected

