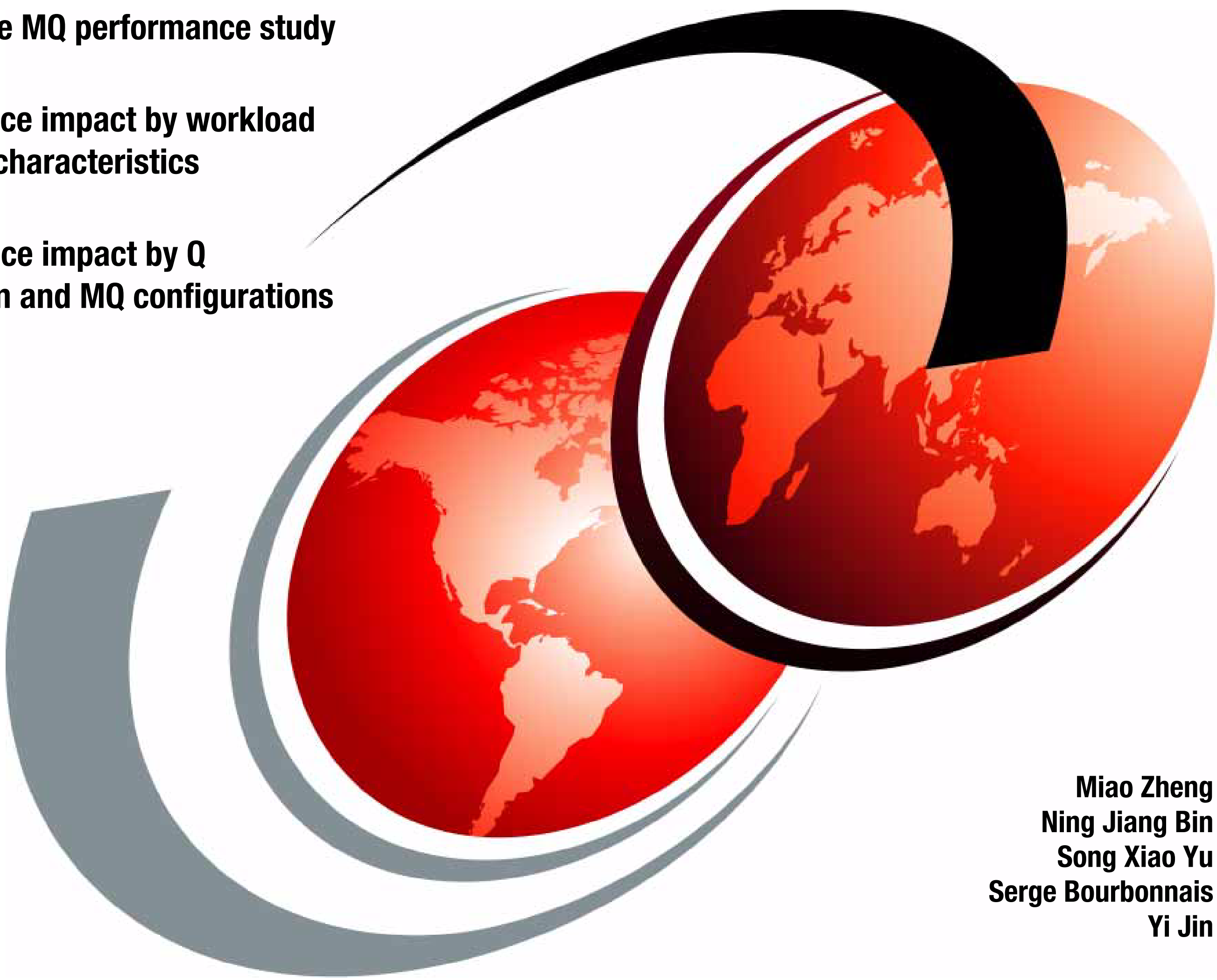
IBM® Information Management Software

# InfoSphere Data Replication for DB2 for z/OS and WebSphere Message Queue for z/OS

## Performance Lessons

- **Q Replication for DB2 for z/OS and WebSphere MQ performance study**
- **Performance impact by workload and table characteristics**
- **Performance impact by Q Replication and MQ configurations**




Miao Zheng
Ning Jiang Bin
Song Xiao Yu
Serge Bourbonnais
Yi Jin


**Redpaper**

ibm.com/redbooks

International Technical Support Organization

# InfoSphere Data Replication for DB2 for z/OS and WebSphere Message Queue for z/OS

## Performance Lessons

December 2012

REDP-4947-00

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

**First Edition (December 2012)**

This edition Version 10, Release 1 of InfoSphere Data Replication for DB2 for z/OS (program number 5655-DRP) and Version 7, Release 0, Modification 1 of WebSphere MQ for z/OS (program number 5655-R36).

# Contents

# Notices

This information was developed for products and services offered in the U.S.A.

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, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** 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 states 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 supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

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 the names and addresses used by an actual business enterprise 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.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICS®
DB2®
GDPS®
IBM®
InfoSphere®
MVS™
OMEGAMON®
Redbooks®
Redpaper™
Redbooks (logo) ®
RMF™
WebSphere®
z/OS®

The following terms are trademarks of other companies:

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

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

Understanding the impact of workload and database characteristics on the performance of both DB2®, MQ, and the replication process is useful for achieving optimal performance. Although existing applications cannot generally be modified, this knowledge is essential for properly tuning MQ and Q Replication and for developing best practices for future application development and database design. It also helps with estimating performance objectives that take these considerations into account.

Performance metrics, such as rows per second, are useful but imperfect. How large is a row? It is intuitively, and correctly, obvious that replicating small DB2 rows, such as 100 bytes long, takes fewer resources and is more efficient than replicating DB2 rows that are tens of thousand bytes long. Larger rows create more work in each component of the replication process. The more bytes there are to read from the DB2 log, makes more bytes to transmit over the network and to update in DB2 at the target.

Now, how complex is the table definition? Does DB2 have to maintain several unique indexes each time a row is changed in that table? The same argument applies to transaction size: committing each row change to DB2 as opposed to committing, say, every 500 rows also means more work in each component along the replication process.

This Redpaper™ reports results and lessons learned from performance testing at the IBM® laboratories, and it provides configuration and tuning recommendations for DB2, Q Replication, and MQ. The application workload and database characteristics studied include transaction size, table schema complexity, and DB2 data type.

## The team who wrote this paper

This Redpaper was produced by a team of specialists from around the world working at the China Development Laboratory, Beijing, China.

**Miao Zheng** is an Architect at the IBM China Development Lab (CDL) zAIM Solution Center and the chair of the CDL z community technical leadership team. Miao joined IBM in 2003 and worked in DB2 for z/OS®, DB2 Tools, CICS®, and MQ product teams. Miao has rich experience in DB2 performance tuning, Q Replication performance in GDPS® Active-Active solution, and is currently the lead of a solution team for developing end2end solutions based on CICS, MQ, WebSphere® on z/OS, and WODM.

**Ning Jiang Bin** joined IBM in 2011 when he graduated from Peking University. He is currently a Software Engineer in the IBM China Development Lab zAIM Solution Center and works in the area of WebSphere MQ for z/OS performance testing and tuning. He provides support to large customers in the finance industry.

**Song Xiao Yu** is a Staff Software Engineer for IBM in China. She has four years developing and testing IBM WebSphere Adapter products. She currently supports large finance customers on z/OS performance tuning and solution design.

**Serge Bourbonnais** is a Senior Technical Staff Member and the Lead Architect for InfoSphere® Replication Server development at the IBM Silicon Valley laboratory in San Jose, California. He was the development lead at the origin of Q Replication.

**Yi Jin** holds a Ph.D degree from Beijing University of Aeronautics and Astronautics, China. He joined the IBM China Development Lab in 2009 and worked in the IBM DB2 for LUW development team as a Compiler and Runtime Developer for three years. He now works in the IBM System z brand team and plays a role of technical and technology enablement in the GDPS/Active-Active area.

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

Paolo Bruni
International Technical Support Organization

Akiko Hoshikawa
Ping Li
David Tolleson
Percy Yu
IBM Silicon Valley Lab

Tony Sharkey
Pete Siddall
IBM Hursley Lab

## 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 ITSO 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

## Comments welcome

Your comments are important to us!

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

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

  **ibm.com**/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks
- Follow us on Twitter:

  http://twitter.com/ibmredbooks
- Look for 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

# 1

# Introduction

Ever since its initial release in 2004, Q Replication has delivered high-performance replication for DB2 for z/OS and on Linux, UNIX, and Windows. It provides low-latency replication, often sub-second, and minimal network traffic using a compact message protocol, and high-volume throughput, generally in the tens of thousands of row changes per second, or even hundreds of thousands.

Q Replication supports a multitude of replication topologies with multidirectional replication; however, the performance results obtained for a unidirectional configuration are applicable to other configurations, except for column suppression, which can also be used for bidirectional configuration but requires running an ignore/ignore conflict resolution mode. For this study, all tests were conducted with a unidirectional configuration.

Our unidirectional configuration is intended for running an application in "active-standby" mode. An active-standby application can be set up with either a bidirectional or a unidirectional configuration.

With unidirectional, replication must be established in the reverse direction, at failover, before the application switch. This configuration is sometimes used if switch-overs are infrequent.

With bidirectional, replication runs all of the time in both directions. However, there are no changes to replicate from the standby site, so replication is generally mostly idle at that site. Discussing multidirectional configurations is outside of the scope of this paper.

For more information, refer to:

http://publib.boulder.ibm.com/infocenter/iidr/v10r1m2/index.jsp

This chapter contains the following topics:

- Background information
- Observations and conclusions

## 1.1 Background information

The IBM Q Replication component of the IBM InfoSphere Data Replication product is IBM strategic technology for deploying DB2 for z/OS active-active configurations across unlimited distance for continuous business availability throughout both planned and unplanned outages. Q Replication is a high performance log capture/ transaction-replay replication technology that uses IBM WebSphere MQ message queues to transmit and stage data between source and target database subsystems. It replicates DB2 transactions asynchronously, not impacting application response time. Q Replication is used to maintain a near real time hot failover site, where applications can be switched during planned or unplanned component or system outages.

An Active/Active solution provides continuous availability throughout:

- Regularly-scheduled maintenance that includes system, database, application upgrades, migrations, and deployments. Because source and target do not need homogeneous configurations, Q replication is often used to combine several upgrades that include hardware and database configuration or version changes.
- Physical recovery from unexpected outages, such as natural disasters or hardware failures.
- Performance degradation at a site where an application runs. Q Replication is often used to direct queries against a near real time copy of the data where they cannot create contention with online business transactions. It is also used to temporarily redirect application workloads while troubleshooting performance problems at their primary site.

An application can be switched to another site, until the planned or unplanned disruption is over.

Continuous availability is achieved when the end user is unaware that a switch took place, which generally requires a Recovery Time Objective (RTO) of seconds. The recovery time includes detecting the failure and switching the application. For planned fail over, it includes the time to finish applying changes in transit and then switching the application. Therefore, replication performance is critical for achieving continuous availability. If replication falls behind, the recovery time is elongated because you must wait until replication catches up before the applications can be switched. With high-volume transactional systems, particularly on the mainframe, replication must keep up with throughputs that are often in the hundreds of millions of changes per day without falling behind.

While default configuration parameters are adequate for a majority of applications, a handful of parameters are particularly sensitive to the specificity of the application workload and database characteristics. This paper reports on a study of the factors that most impact Q Replication and MQ performance for varying database and workload characteristics.

Understanding the impact of workload and database characteristics on the performance of both DB2, MQ, and the replication process, is useful for achieving optimal performance. Existing applications cannot generally be modified, but this knowledge is essential for properly tuning MQ and Q Replication and developing best practices for future application development and database design.

## 1.2 Observations and conclusions

The MQ message size plays a significant role in the performance of Q Replication. Very small messages cause a bottleneck at the channel level, but this can be alleviated by using trans_batch_size. Any tuning that can contribute to batching in DB2 (such as larger DB2 transactions as opposed to committing every row) reduces I/O, minimizes overhead, and improves the MQ performance. A goal of performance tuning is achieving a balance where the MQ buffer pool hit ratio is high, and MQ read-ahead is effective. Best performance results are obtained with messages from 32 KB to 256 KB in size. This range provides optimal MQ performance.

Having a large number of columns in DB2 tables is not ideal for performance. It is better to add them when needed rather than defining them as placeholders for future use.

Q Replication supports automatic replication of alter add column operations. However, good performance can be achieved for tables with large numbers of columns, even several hundred, with Q Replication column suppression, which can be used when a conflict_action of Ignore meets the business requirements.

TRANS_BATCH_SZ and Q Apply commit_count both contribute to larger DB2 transactions at the target, reducing I/O and further improving performance. They are particularly recommended for online workloads where DB2 transactions are small, such as 1-2 rows per DB2 commit, as long as the batching does not introduce contention between batched transactions.

Row-level locking at the target is highly recommended. Our tests and the experience of several customers indicate that it does not have a detrimental effect on system performance, especially for read/standby systems. The major impact/overhead of row level locking is the additional page p-lock it introduced in the DB2 data sharing environment. Certain customers deploy procedures to alter table space locking back to page-level if and when they failover an application to the target.

The MQ read-ahead new feature can significantly improve the performance when reading messages from page set. Because the maximum buffer pool size is ~1G, in a real production system with huge batch jobs running, MQ might not be able to hold all of the messages in the buffer pool, so read-ahead is extremely useful to catch up on the backlog.

# 2

# Environment and scenario

This chapter contains the following topics:

- Objectives
- Test methodology
- Q Replication metrics
- Q Replication and MQ parameters
- Test results using the initial configuration

## 2.1 Objectives

The objective of this paper is to understand the most significant factors that impact end-to-end performance of Q Replication for DB2 for z/OS for high volume environments.

Because the overall performance of any system is determined by its slowest component, and because bottlenecks tend to shift from one component to another under various conditions, we study the performance of the major individual components of the replication process. Figure 2-1 illustrates the potential bottlenecks in the replication process. Factors, which include workload characteristics, CPU, disk access speed, DB2, MQ, and Q Replication tuning and configuration, determine which bottleneck might be encountered for a given installation.

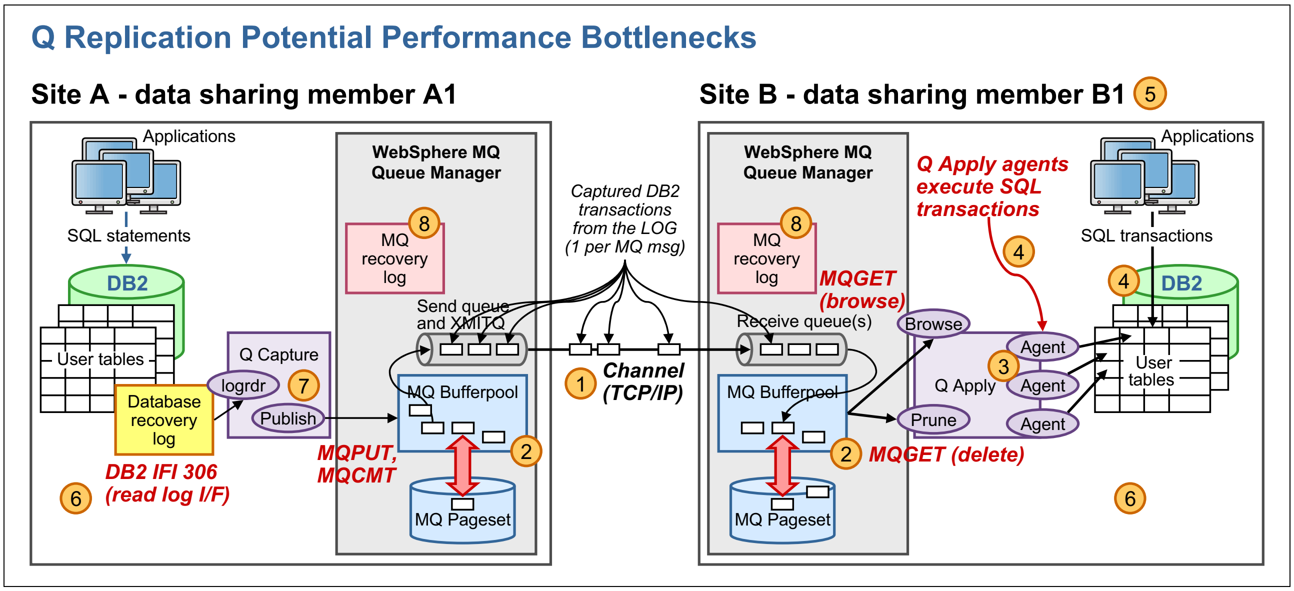



*Figure 2-1 Q Replication potential performance bottlenecks*

In Figure 2-1:

1. MQ channel (TCP/IP connection): Message size influences performance. Smaller messages can limit maximum throughput.
2. MQ buffer pool: The buffer pool can fill up under high volume, which causes spilling to a page set. Reading or writing to the page set can reduce replication throughput.
3. Q Apply serialization: It can be caused by DB2 transactions that update the same row.
4. DB2 lock contention: It can be caused by other applications accessing the target database or by multiple apply agents applying transactions in parallel. Lock contention is potentially worse with page-level locking, particularly during some massive batch jobs, where a high-level of parallelism in Q Apply can result in transactions requesting locks for the same page. When this occurs, row-level locking at the target is recommended, or parallelism must be reduced by using the Q Apply MAXAGENTS_CORRELID parameter.
5. Q Apply program limited to a single data sharing member: Q Apply ensures dependent transaction consistency at all times for a set of tables defined as Consistency Groups. This requires running in one address space. Note that applications that use independent sets of tables can each have a dedicated Q Apply program, which can each be fed by the same Q Capture program at the source. This provides scalability beyond the limit of a single data-sharing member.

6. z/OS CPU and DISK I/O speed for both DB2 and MQ data.
7. Q Capture thread publishing to MQ can become CPU constrained: This thread includes MQ PUT and COMMIT elapsed times. When the bottleneck moves here, it is often related to factor number 6, CPU resources and DISK I/O speed.
8. MQ logging for persistent messages: Maximum achievable logging rate depends on message size and I/O speed.

# 2.2 Test methodology

At a high-level, the performance bottlenecks roughly fall between disk and CPU for DB2, MQ, and Capture/Apply.

To better understand these bottlenecks, we measured and reported the following performance metrics:

- Q capture and Q apply throughput and latency, obtained from the Q Apply monitor table
- MQ channel throughput, obtained by running the MQ monitor program.
- MQ accounting data, such as MQGET/MQPUT elapsed time, and log rate, obtained from the report formatted by MQ SupportPac MP1B.
- CPU time for Q capture and Q apply, obtained from RMF™ workload activity report.
- CPU utilization for each LPAR, obtained from RMF CPU report.
- Target DB2 elapsed time for Q apply, obtained from IBM Tivoli OMEGAMON® XE for DB2 Performance Monitor on z/OS.

These metrics are useful but imperfect. It is intuitively, and correctly, obvious that replicating small DB2 rows, such as only 100 bytes long, is faster than replicating DB2 rows that are tens of thousands of bytes long because it causes more work in each component of the replication process. That is, larger rows mean more bytes to read from the DB2 log, more bytes to transmit over the network, and more bytes to update in DB2 at the target. The same obvious consideration applies to transaction size: committing each row change to DB2 as opposed to committing, say, every 500 rows is more work in each component of the process. For this reason, we studied the impact of row and transaction size on the replication performance metrics.

## 2.2.1 Q Replication metrics

The Q Replication technology provides a set of useful tables for performance monitoring that are updated at each monitor_interval by the respective Q Capture and Q Apply programs. See Figure 2-2 on page 8. These tables are the IBMQREP_CAPMON (for Q Capture program metrics), IBMQREP_CAPQMON (for send queue metrics), and IBMQREP_APPLYMON (for Q Apply receive queue metrics). For this test, we set the intervals to 10 seconds, that is, each 10 seconds, the Q Capture and Q Apply programs insert a row into these tables with live performance metrics. Q Replication monitoring can be used at all times, without any measurable overhead, even with intervals as low as 1 second.

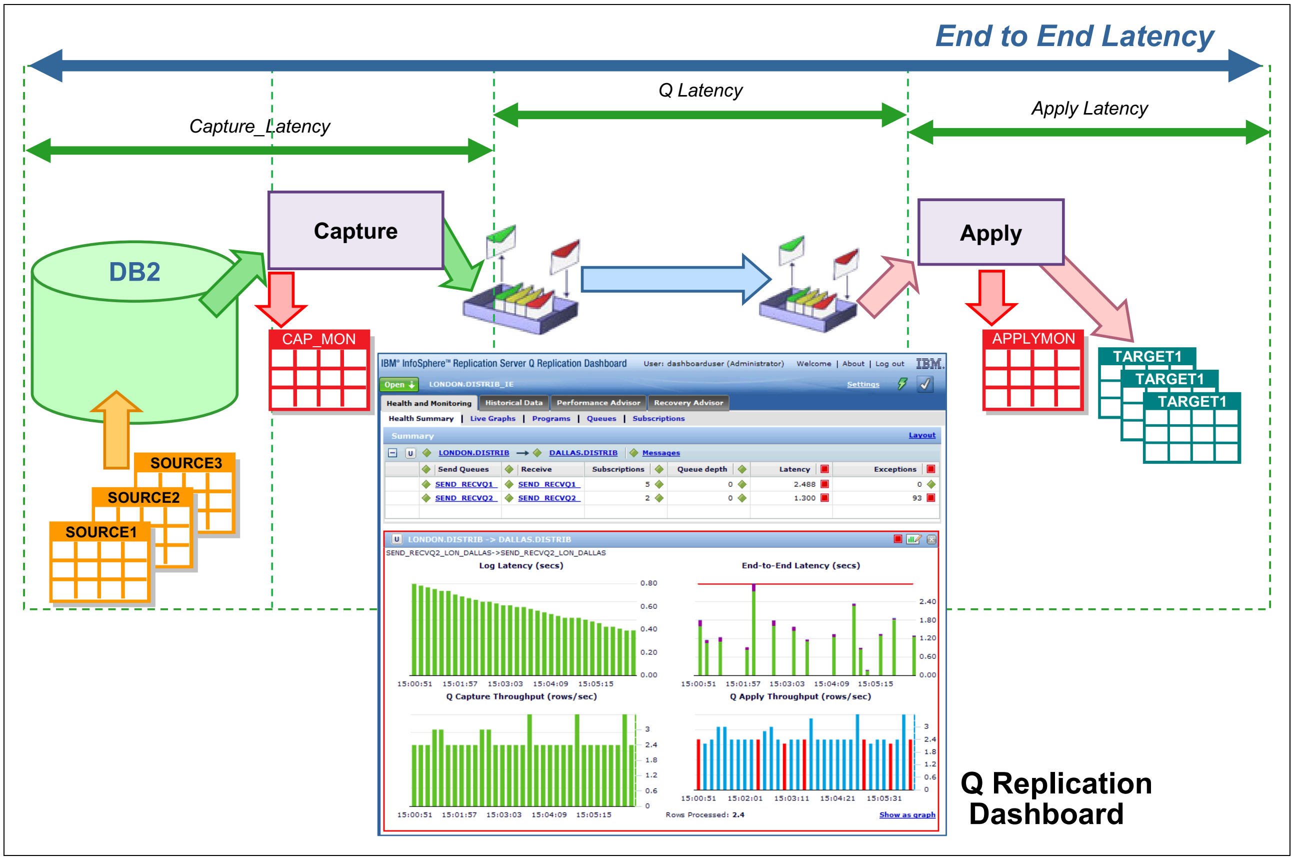



*Figure 2-2   Q Replication latencies reported in the Apply Monitor table IBMQ Replication_APPLYMON*

In Figure 2-2:

- Latency counters report averages for each Q Replication monitor interval:
  - Q Apply adds up latency values for each transaction and divides the final sum by the number of transactions during the monitor interval.
  - All counters are reset at each monitor interval.
  - If there are no transactions applied in the monitor interval, latency is 0.
  - All times are calculated using GMT.
  - Intervals are reported in milliseconds.
- END2END_LATENCY:
  - DB2 transaction commit time at target minus DB2 commit time at the source.
  - Commit time at the source database is from the DB2 commit log record.
  - Commit time at the target database is obtained by getting a time stamp from the system after committing the target DB2 database.
- CAPTURE_LATENCY:
  - MQPUT time of last message used to send the DB2 transaction minus source commit time.
  - MQPUT time is time stamp generated by MQ.
  - Commit time at the source database is from the DB2 commit log record.

- APPLY_LATENCY

  DB2 transaction commit time at target minus MQGET time in first message that delivers the database transaction

- QLATENCY

  Computed by Q Apply as:

  `END2END_LATENCY - CAPTURE_LATENCY - APPLY_LATENCY`

  This is effectively the time spent in MQ, both transmission and staging, and therefore not necessarily a reflection of network transmission speed.

### 2.2.2 MQ metrics

MQ provides the following metrics:

- MQSC[1] command

  MQ provides the MQSC command **DISPLAY CHSTATUS (channel-name) ALL** to display the channel status. It displays the messages and bytes sent during the session. We use a program to issue this command every 10 seconds and format the results. In this way, we can get the channel performance data in real time. MQSC command **RESET QSTATS(RECVQ.A1.A2)** can be used to monitor MQ queue, it displays the peak queue depth and messages put or get from the queue during the interval.

- MQ SupportPac MP1B

  MQ SupportPac MP1B provides information about the use and interpretation of the new accounting and statistics available in MQ for z/OS Version 7.0. Also supplied are programs (including one written in C and JCL) that can be used to display the data. We use this program to get the MQPUT/MQGET elapsed time and log rate.

### 2.2.3 CPU and DB2 metrics

We define a report class for Q Capture and Apply. With the RMF workload activity report, Figure 2-3 on page 10, we know the CPU cost by Q Capture and Apply; therefore, we can calculate the CPU cost for replicating each row.

The RMF interval is five minutes. Each test runs longer than 10 minutes to have at least one RMF CPU interval for our tasks.

[1] MQ SCript commands

```
CPU time is from the WLM report class for the Q capture and apply address space: CPU+SRB
SERVICE TIME ---APPL %---
CPU 111.973 CP 37.33
SRB 0.007 AAPCP 0.00
RCT 0.000 IIPCP 0.00
IIT 0.000
HST 0.000 AAP N/A
AAP N/A IIP N/A
IIP N/A
LPAR/MVS BUSY is from CPU report, we are using dedicated CP, so the they are the same
---CPU--- ---------------- TIME % ---------------- LOG PROC
NUM TYPE ONLINE LPAR BUSY MVS BUSY PARKED SHARE %
0 CP 100.00 30.46 30.46 ------ 100.0
1 CP 100.00 30.38 30.38 ------ 100.0
2 CP 100.00 30.31 30.31 ------ 100.0
TOTAL/AVERAGE 30.38 30.38 300.0
```

*Figure 2-3 RMF workload activity report*

The OMEGAMON PE accounting report is used to get DB2 CLASS 2 elapsed time and CLASS 3 suspensions.

# 2.3 Test scenario description

The scenarios were devised to answer the following questions:

- How do application workloads and database characteristics impact performance?
- How to configure and tune MQ and Q Replication taking such characteristics into account?

## 2.3.1 Application configuration

The use-cases studied in this paper covered:

- Workload and database characteristics:
  - Number of columns in the table: 12, 30, 60
  - Data types: INT / DECIMAL / CHAR
  - Transaction size: 5, 10, 50 rows per commit
  - Number of indexes on the table: 1, 3, 5 indexes per table
- Q Replication configuration:
  - Column suppression: on, off
  - TRANS_BATCH_SZ: 1, 3, 5
  - Number of apply agents: 1, 4, 6, 8, 12,16
  - MEMORY_LIMIT in Q capture: 200 MB, 500 MB, 800 MB
  - COMMIT_INTERVAL setting: 200 ms, 500 ms, 800 ms
  - MEMORY_LIMIT setting in Q apply: 100 MB, 200 MB,
- MQ configuration:
  - Message persistence: yes, no
  - Buffer pool read-ahead: on, off
  - MQ GLOBAL/CHINIT trace: on, off
  - MQ security: on, off
  - Shared queues, non-shared queues

## 2.3.2 Environment configuration

This section provides the hardware, software, and system set up used in this test:

- Hardware configuration:
  - z196 2817-778 model M80
  - 1 SYSPLEX with 3 LPARs
  - Each LPAR with 3 dedicated CPs
- Software configuration:
  - z/OS V1R11
  - DB2 9 for z/OS
  - Q Replication V10.1.0 APAR PM63905
  - MQ V701 APAR PM63802
- Test configuration

  Figure 2-4 shows the configuration that was used for producing this paper. The workload runs on LPAR KA, which is the source. Q Capture and the source queue manager run in LPAR KB so as not to compete for resources with the source workload. Another queue manager runs with Q Apply at the target LPAR KC. Messages are transmitted across a single channel. There is no significant network delay because the source and target are on the same SYSPLEX. We suggest using a dedicated MQ queue manager for Q Replication to achieve best performance and define different page sets for different usage purpose.

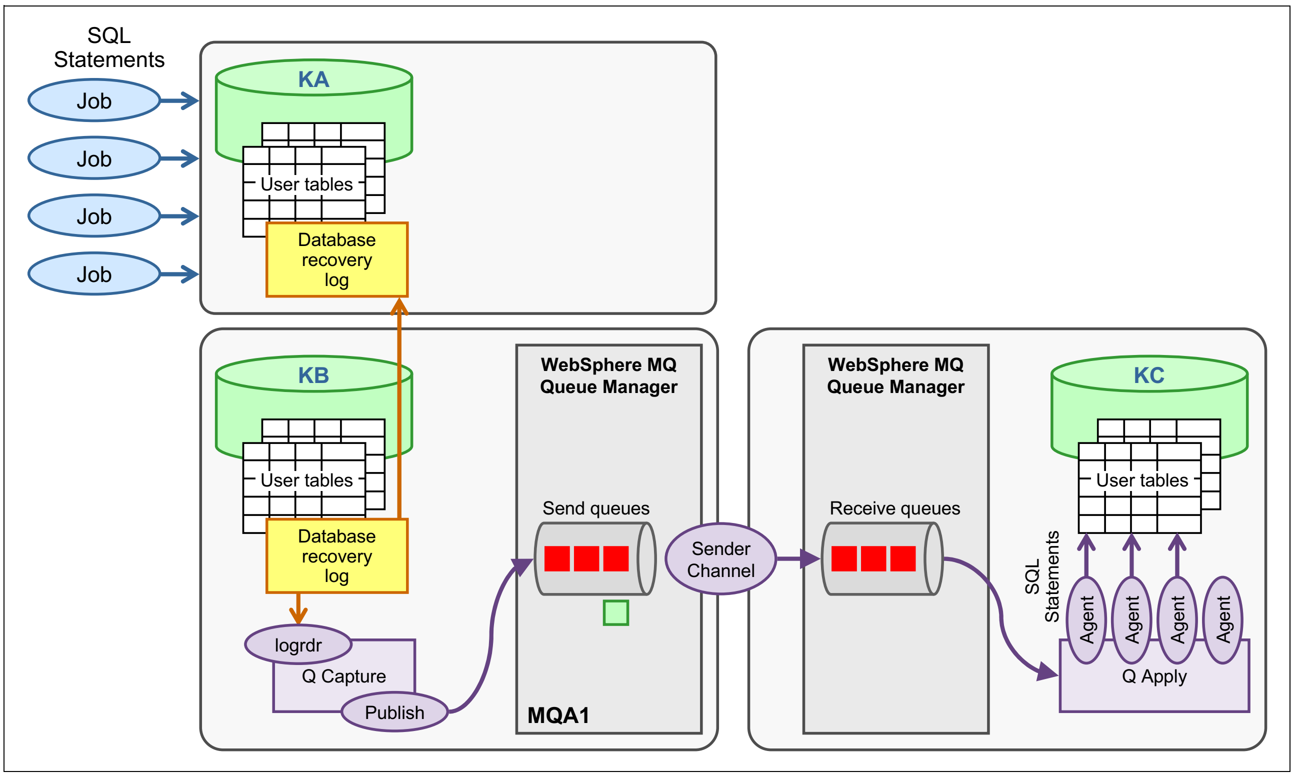



*Figure 2-4 Test configuration*

## 2.4 Q Replication and MQ parameters

We started with an initial configuration already optimized for an existing customer, based on their workload. Tuning for that initial configuration already changed default values for the following default MQ, DB2, and Q Replication settings, for the reasons described in Table 2-1. Some parameters, such as TRANS_BATCH_SZ, were already studied in production, and set to 3 for optimizing MQ message size and DB2 performance. We expand in more details about these decisions further in this paper. We also explain the lessons learned tuning the system for performance under different variables. For each test scenario, one (and only one) parameter is changed to different values, so its impact can be measured.

Table 2-1 lists the Q Capture variables.

*Table 2-1 Q Capture*

| Parameter | Value | Justification |
|---|---|---|
| MEMORY_LIMIT | 500 MB | Sufficient for all workload, as measured by CAPMON current_memory counter. |
| COMMIT_INTERVAL | 500 ms | Q Capture commits MQ after a maximum of 500 ms. However, at high throughput, the maxtrans limit is reached much sooner and Capture commits much more frequently, less than every 50 ms. |
| SLEEP_INTERVAL | 50 ms | This was established as being optimal during our performance test, yielding more predictable latency throughout irregular workload. |
| TRANS_BATCH_SZ | 3 | In our initial tests, workload was OLTP with small DB2 transactions (5 rows changed per commit), resulting in small MQ messages and a bottleneck at the MQ channel level. Batching them in groups of 3 resulted in larger transactions, without introducing any contention for the workload studied. |
| MSG_PERSISTENCE | Y (default) | Recommended setting for continuous availability scenarios. |
| MONITOR_INTERVAL | 10000 ms | 10 seconds limits the amount of data produced, but tests with a 1 second interval were also carried out without any issue. |
| MAX_MESSAGE_SIZE | 10240 KB | For this environment, the largest single DB2 transaction never exceeds 10 MB. This parameter can be set to any value to accommodate the largest DB2 transactions and avoid spilling. This value must align with the MQ MAXMSGL setting. |
| CHANGED_COLS_ONLY | N | This is required for using **F** (force) conflict resolution. All column values must be sent so that an UPDATE can be transformed into an INSERT if and when the row is not found at the target because of a conflicting application change in an Active/Active configuration. |

Table 2-2 lists the Q Apply variables.

*Table 2-2 Q Apply*

| Parameter | Value | Justification |
|---|---|---|
| NUM_APPLY_AGENTS | 6 | Reduced to maintain CPU below 90% at the target on the initial system, which had limited CPU capacity. |
| MEMORY_LIMIT | 200 MB | Adequate for all workload. Larger values have no impact. You can allocate as much as available. |

| Parameter | Value | Justification |
|---|---|---|
| MONITOR_INTERVAL | 10000 ms | The monitor_interval sets the frequency of Q Apply inserts into the IBMQREP_APPLYMON table. 10 seconds is a good balance of granularity and cpu overhead. |
| CONFLICT_ACTION | F | If conflict happens on the Apply side, force the action. |

Table 2-3 lists the MQ variables.

*Table 2-3 MQ*

| Parameter | Value | Justification |
|---|---|---|
| BATCHSZ | 200 | It defines the maximum number of messages to be transferred between two syncpoints. A large value for the batch size increases throughput, but recovery times are increased because there are more messages to back out and re-send. We set this value higher than the number of messages Q Replication put to MQ queue in one commit. |
| MAXMSGL | 20971520 bytes | This is only a maximum. Performance considerations are only for the actual message size transmitted, not the maximum. Just make sure this parameter is larger than the value of MAX_MESSAGE_SIZE in Q Replication (Q Replication issues a warning if it is not larger), which you use for tuning the message size. |
| MAXSHORTMSGS | -1 | MAXSHORTMSGS(0) means that all small messages are candidates for multiple messages on a 4K page. This is good for capacity purposes, after the message is retrieved from the queue, it is difficult for the queue manager to know when the page is available for re-use because there might still be other messages on the queue. We need additional overhead to scan the pages and mark those available for re-use.<br>MAXSHORTMSGS(-1) means that all small messages will be 1 message per 4K page. This saves the additional work to scan pages. |
| Security | OFF | Disabled all MQ security by RDEFINE MQADMIN <QMGR>.NO.SUBSYS.SECURITY. |
| READ-AHEAD | ON | New buffer pool read-ahead function is provided to pre-stage messages from deep queues into the MQ buffer pool to improve performance of Q Replication. It is available in PM63802/ UK79853. |
| LOGLOAD | 16000000 | It specifies the number of log records that MQ writes between the checkpoints. The range 200 through 16,000,000 and default is 500,000. The greater the value, the better the performance of MQ; however, restart takes longer if the parameter is set to a large value. |
| BUFFPOOL | 240000 (4 KB pages) | The MQ buffer pool can be the most critical factor for MQ performance. Ideally, it is large enough for keeping any backlog of messages in memory. After it fills, messages go to disk. For optimal performance, IBM recommends a dedicated Q manager for Q Replication usage. In our configuration, we use a single queue between the source and target and allocate the maximum possible buffer pool for this queue. The associated page set is also dedicated for the Q Replication queue. |
| Trace | ACCTG(3), STAT(*) | This trace introduces no significant overhead on MQ and is essential for performance test analysis. |

| Parameter | Value | Justification |
|---|---|---|
| PAGESET setting | PSID00 | MQ object |
| | PSID01 | System queues |
| | PSID02 | Long live message |
| | PSID03 | Volatile queue (admin and restart queue) |
| | PSID04 | Q Replication queues |

The workload is generated by invoking native SQL stored procedures that perform insert/update/delete statements. Scalability is achieved by running multiple threads, each thread calling one invocation of the stored procedure.

Unless explicitly specified in that test, the default table schema used for testing is:

- Column number: 30
- Column types:
  - 10 INTEGER
  - 10 CHAR(20):
    - 5 DATE
    - 5 DECIMAL(20,0)
- Index number: 1
- Index type: INTEGER

All columns are defined with DEFAULT to NULL. The INSERT statement inserts values for all columns. The UPDATE statement updates one column. The workload updates different rows so that artificial dependencies are not introduced.

The workload generator can vary the number of columns modified by each update, the number of rows changed in a transaction, and the ratio of insert/update/delete. Unless it is explicitly specified for that test, the default workload is:

- Transaction size: 5 rows with 1 insert, 3 update, 1 delete
- Update statement: update 3 columns (int, char, date)

## 2.5 Test results using the initial configuration

Table 2-4 lists the maximum throughput and latency achieved using the initial configuration. These results are used as a baseline for measuring test result impacts.

The workload used reflects an OLTP system with small DB2 transactions, which is not necessarily optimal for performance, which we previously explained.

*Table 2-4 Performance data in baseline test*

| Test # | Baseline |
|---|---|
| Capture throughput (rows/sec) | 43.4K |
| MQ average message size | 5.8 KB |
| MQ average channel throughput (messages/sec) | 2910 |
| MQ average channel throughput (MBps) | 16.9 MB |

| Test # | Baseline |
|---|---|
| Q Apply throughput (rows/sec) | 43.2 K |
| Q Replication END2END_LATENCY (ms) | 828 |
| MQ QLATENCY (ms) | 43 |
| Q Apply APPLY_LATENCY (ms) | 219 |
| Q Capture CAPTURE_LATENCY (ms) | 566 |
| Q Capture LPAR MVS™ BUSY | 30 |
| Q Apply LPAR MVS BUSY | 83 |
| Q Apply DB2 class 2 elapsed time (ms) | 1.95 |

The chart in Figure 2-5 shows the performance behavior in the baseline test. The throughput is stable with low latency.

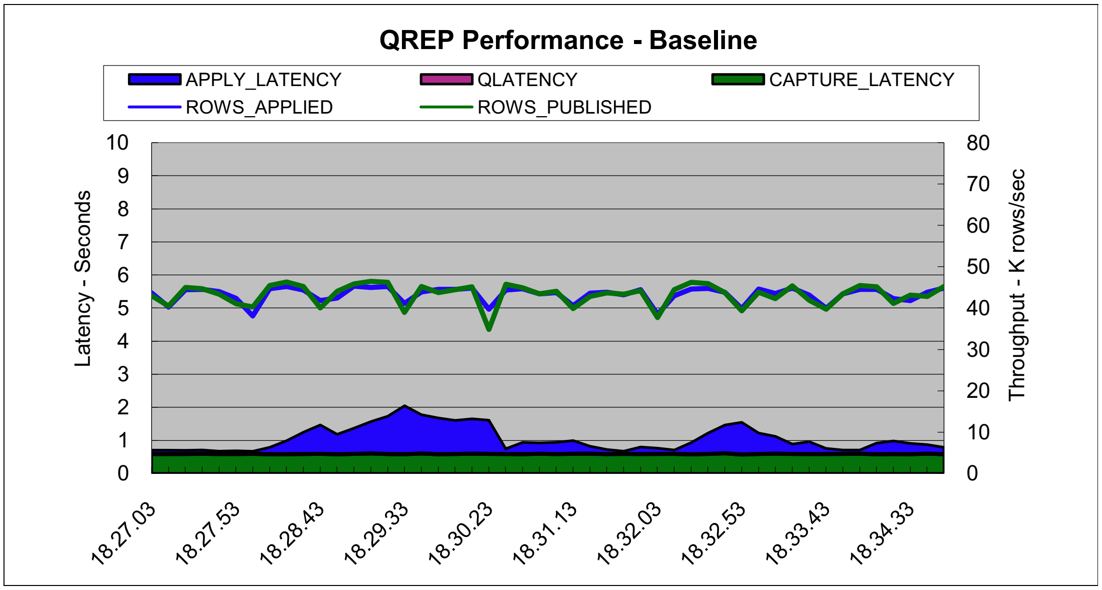



*Figure 2-5 Q Replication performance in baseline test*

These results are excellent. The initial configuration for the OLTP workload gave a sub-second end-to-end latency and a stable and sustained throughput over 40K rows/second. As a matter of fact, in this environment, the workload cannot be driven to exceed the speed of replication.

The initial configuration is optimal for this particular OLTP workload, but a production environment sees a wide range of different workloads. Increasingly, we see a mixture of large batch jobs and online workloads, which requires balancing the configuration/tuning for the more general case and some compromises. This is why we studied the impact of workload variations to achieve a configuration that is optimal for mixed workloads.

# 3

# Impact of variations in workload and table characteristics

In this chapter, we discuss the impact of the number of columns defined in a table, the number of columns modified in update statements, data types, transaction size, and number of indexes.

This chapter discusses the following topics:

- Number of columns
- Data types
- Transactions size
- Number of unique indexes

## 3.1 Number of columns

In this scenario, we compare the throughput using tables with different numbers of columns: 12, 30, and 60, trying to keep the same record length. See Table 3-1:

- The column types for 12 columns: 4 int, 4 char(67), 2 date, 2 decimal(20,0), row size=314 bytes
- The column types for 30 columns: 10 int, 10 char(20), 5 date, 5 decimal(20,0), row size=315 bytes
- The column types for 60 columns: 20 int, 20 char(4), 10 date, 10 decimal(20,0), row size=310 bytes

*Table 3-1 Performance data for different column number*

| Test # | 12 columns | 30 columns | 60 columns |
|---|---|---|---|
| Capture throughput (rows/sec) | 44.5K | 43.4K | 40K |
| Message size | 5.2 KB | 5.8 KB | 6.8 KB |
| Channel throughput (messages/sec) | 2997 | 2910 | 2712 |
| Channel throughput (MBps) | 15.5 MB | 16.9 MB | 18.3 MB |
| Q Apply throughput (rows/sec) | 44.4K | 43.2K | 40K |
| END2END_LATENCY (ms) | 812 | 828 | 836 |
| QLATENCY (ms) | 45 | 43 | 48 |
| APPLY_LATENCY (ms) | 205 | 219 | 219 |
| CAPTURE_LATENCY (ms) | 561 | 566 | 568 |
| Q Capture LPAR MVS BUSY | 26 | 30 | 28 |
| Q Apply LPAR MVS BUSY | 83 | 83 | 81 |
| Q Capture CPU cost per row (µs) | 12 | 14.7 | 16.5 |
| Q Apply CPU cost per row (µs) | 40 | 42.4 | 47 |
| Q Apply DB2 class 2 elapsed time (ms) | 1.917 | 1.95 | 2.0 |
| Throughput improvement | | -2% | -8% |

The chart in Figure 3-1 shows the impact of the number of columns in a table definition on the throughput.

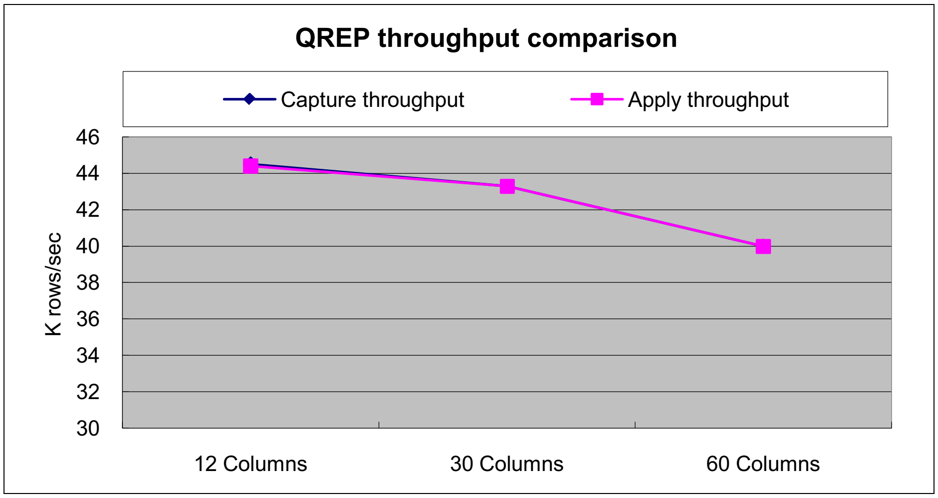



*Figure 3-1 Replication throughput with varying number of columns in the table definition*

The chart in Figure 3-2 shows the CPU usage per row by Q Replication.

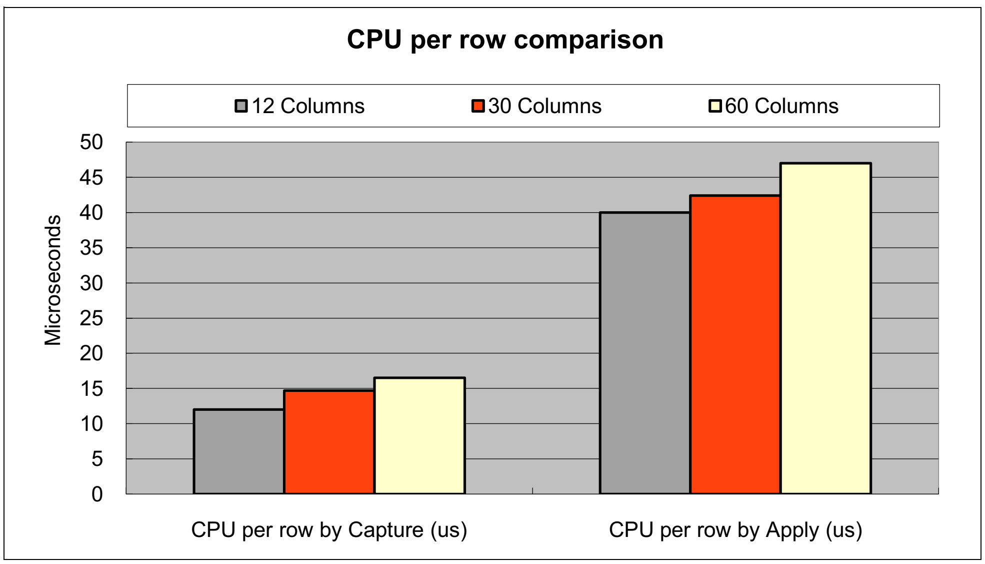



*Figure 3-2 CPU usage by Q Replication for different columns*

A large number of columns can have a significant impact on the maximum achievable rate because of the overhead of processing each column.

## 3.2 Data types

In this scenario, we look at the impact of data types on replication processing. We tested INTEGER, DECIMAL(20,0), and CHAR(20), as shown in Table 3-2 on page 20. All columns in the table are the same type, and the number of the column is 30.

*Table 3-2 Performance data for different column types*

| Test # | INTEGER | DECIMAL | CHAR |
|---|---|---|---|
| Capture throughput (rows/sec) | 47K | 39.6K | 39K |
| Message size | 3.3 KB | 6 KB | 9 KB |
| Channel throughput (messages/sec) | 3178 | 2904 | 2634 |
| Channel throughput (MBps) | 10.2 MB | 16.6 MB | 23.3 MB |
| Apply throughput (rows/sec) | 47K | 39.6K | 39K |
| END2END_LATENCY (ms) | 641 | 1388 | 685 |
| QLATENCY (ms) | 30 | 46 | 65 |
| APPLY_LATENCY (ms) | 46 | 777 | 55 |
| CAPTURE_LATENCY (ms) | 565 | 561 | 563 |
| Q Capture LPAR MVS BUSY | 28 | 27 | 26 |
| Q Apply LPAR MVS BUSY | 85 | 75 | 77 |
| Q Apply DB2 class 2 elapsed time (ms) | 1.76 | 2.14 | 2.03 |

The graph in Figure 3-3 shows a throughput comparison for different column types.

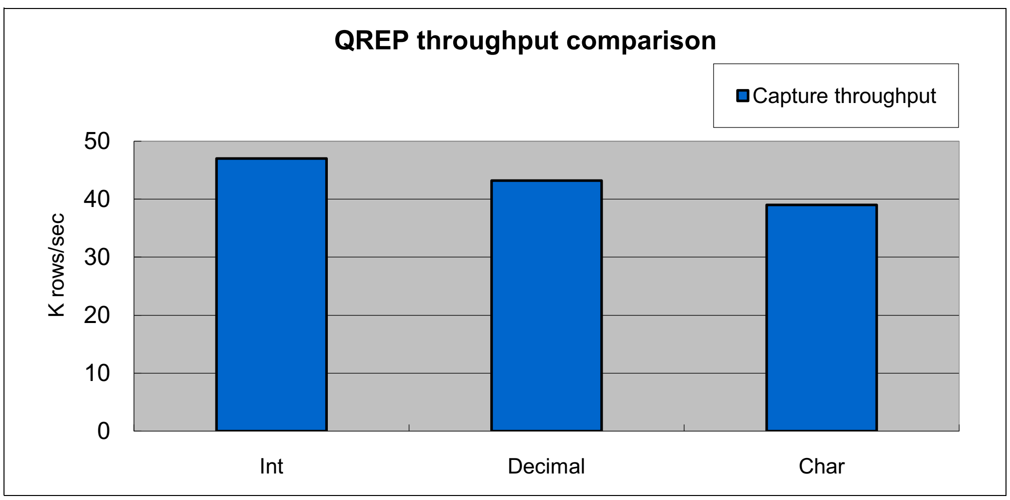



*Figure 3-3 Impact of data type on replication overhead*

The chart in Figure 3-4 on page 21 shows the CPU usage per row by Q Replication.

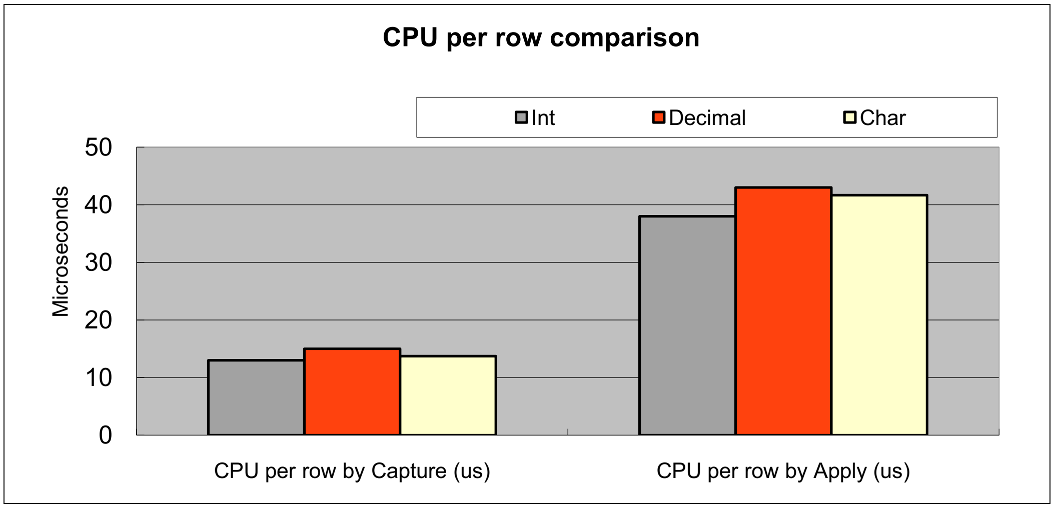


*Figure 3-4 CPU usage by Q Replication for different column types*

Different data types have different processing cost. For example, for decoding, integer is the cheapest and decimal and char are more expensive. There is also the cost of additional code-page related processing for character.

## 3.3 Transactions size

In this scenario, we compare the throughput using different workload transaction sizes: 5, 10, and 50. See Table 3-3:

- Transaction size=5: 1 insert, 3 update (1 int, 1 char, 1 date), 1 delete per commit.
- Transaction size=10: 2 insert, 6 update (1 int, 1 char, 1 date), 2 delete per commit.
- Transaction size=50: 10 insert, 30 update (1 int, 1 char, 1 date), 10 delete per commit.

*Table 3-3 Performance data of using different transaction sizes*

| Test # | Transaction | | |
|---|---|---|---|
| | **size=5** | **size=10** | **size=50** |
| Capture throughput (rows/sec) | 43.3K | 63.7K | 104K |
| Message size | 5.8 KB | 11.2 KB | 54.2 KB |
| Channel throughput (Messages/sec) | 2927 | 2132 | 705 |
| Channel throughput (MBps) | 17 MB | 23.6 MB | 37.4 MB |
| Apply throughput (rows/sec) | 43.3K | 63.7K | 103.5K |
| END2END_LATENCY (ms) | 1051 | 717 | 1430 |
| QLATENCY (ms) | 43 | 63 | 369 |
| APPLY_LATENCY (ms) | 441 | 82 | 474 |
| CAPTURE_LATENCY (ms) | 566 | 570 | 585 |
| Q Capture LPAR MVS BUSY | 30 | 32 | 38 |
| Q Apply LPAR MVS BUSY | 83 | 86 | 95 |

| Test # | Transaction | | |
|---|---|---|---|
| | size=5 | size=10 | size=50 |
| Q Capture CPU cost per row (µs) | 14.7 | 11.3 | 8.6 |
| Q Apply CPU cost per row (µs) | 42.4 | 30.3 | 22.1 |
| Q Apply DB2 class 2 elapsed time (ms) | 1.95 | 2.48 | 7.57 |
| Throughput improvement | | 47% | 63% |

The chart in Figure 3-5 shows the transaction size impact on throughput.

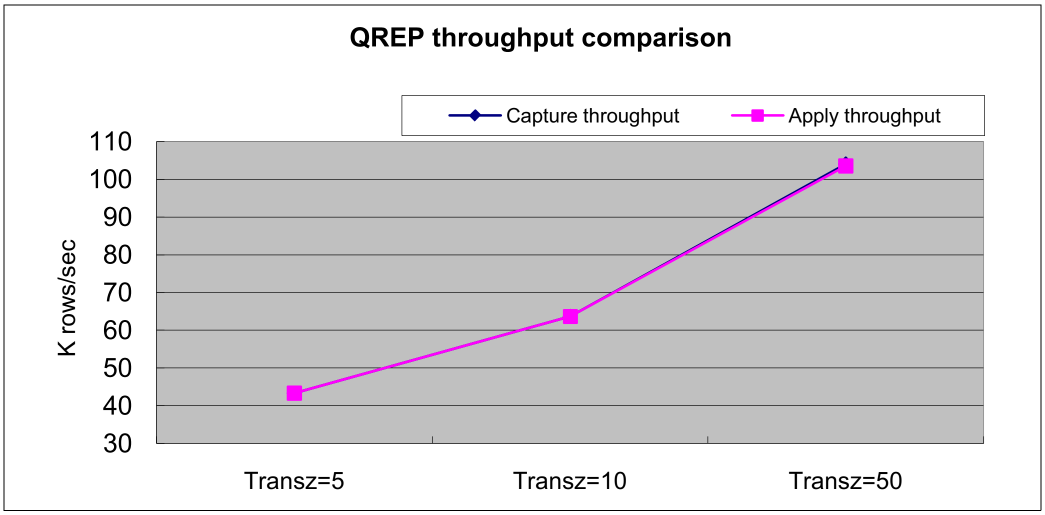



*Figure 3-5 Q Replication with different workload transaction size*

The chart in Figure 3-6 shows the CPU usage per row by Q Replication.

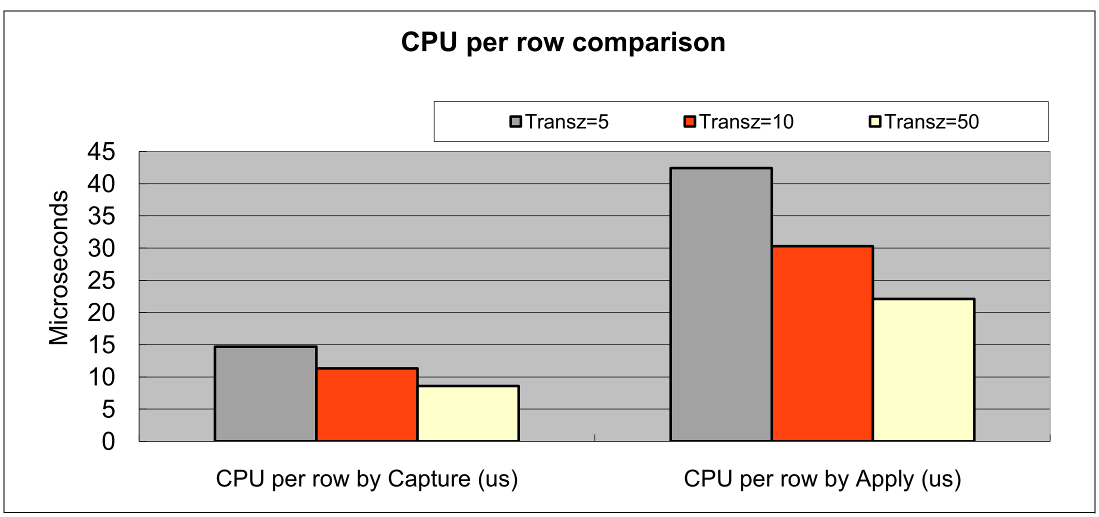



*Figure 3-6 CPU usage per row by Q Replication for different workload transaction size*

Transaction size impacts Replication throughput and CPU. Because Q Replication uses one MQ message for each replicated DB2 transaction, small transactions result in more MQ log writes. Small transactions also involve more overhead with DB2 for transaction processing.

When transaction size increases with more rows per transaction, the less commit points are processed. The reduced commits imply less overhead of transaction processing cost. The impact can be significant for particularly small transactions. The impact becomes less significant when the transaction size increases and mostly negligible when the transaction size is large because the proportion of transaction processing costs become less significant in the total cost.

## 3.4 Number of unique indexes

In this scenario, we compare the throughput using tables with different numbers of unique indexes: 1, 3, and 5.

Unique indexes can affect Q Apply parallel processing because updates to columns that are part of an index can introduce dependencies between transactions, affecting the achievable level of parallelism. Q apply must take all unique constraints into consideration to determine transaction dependencies, which introduces overhead that is proportional to the number of unique indexes. See Table 3-4.

*Table 3-4 Performance data for different number of unique indexes*

| **Test #** | **Number of unique indexes** | | |
|---|---|---|---|
| | **1** | **2** | **5** |
| Capture throughput (rows/sec) | 43.3 K | 36.4 K | 33 K |
| Message size | 5.8 KB | 5.9 KB | 6.1 KB |
| Channel throughput (messages/sec) | 2927 | 2533 | 2226 |
| Channel throughput (MBps) | 17 MB | 15.1 MB | 13.5 MB |
| Apply throughput (rows/sec) | 43.3 K | 37.6 K | 33 K |
| END2END_LATENCY (ms) | 1051 | 659 | 668 |
| QLATENCY (ms) | 43 | 44 | 45 |
| APPLY_LATENCY (ms) | 441 | 53 | 58 |
| CAPTURE_LATENCY (ms) | 566 | 561 | 565 |
| Q Capture LPAR MVS BUSY | 30 | 27 | 26 |
| Q Apply LPAR MVS BUSY | 83 | 84 | 83 |
| Q Capture CPU cost per row (µs) | 14.7 | 15.7 | 17 |
| Q Apply CPU cost per row (µs) | 42.4 | 50.5 | 52 |
| Q Apply DB2 class 2 elapsed time (ms) | 1.95 | 2.2 | 2.5 |
| Throughput improvement | | -13% | -12% |

The chart in Figure 3-7 on page 24 shows throughput reduction with more indexes in the workload table.

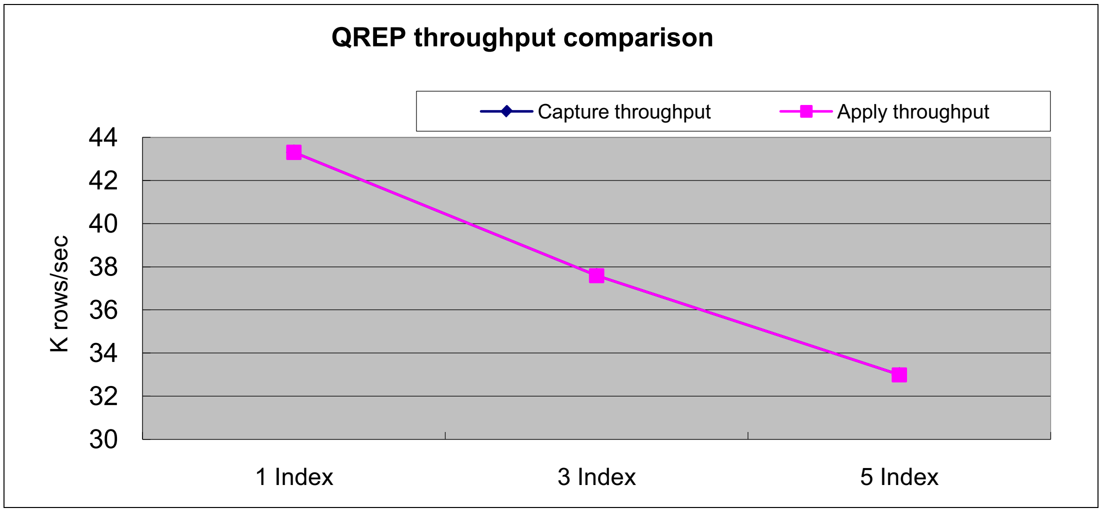


*Figure 3-7 Q Replication throughput with different columns in the index*

The chart in Figure 3-8 shows the CPU usage per row by Q Replication.

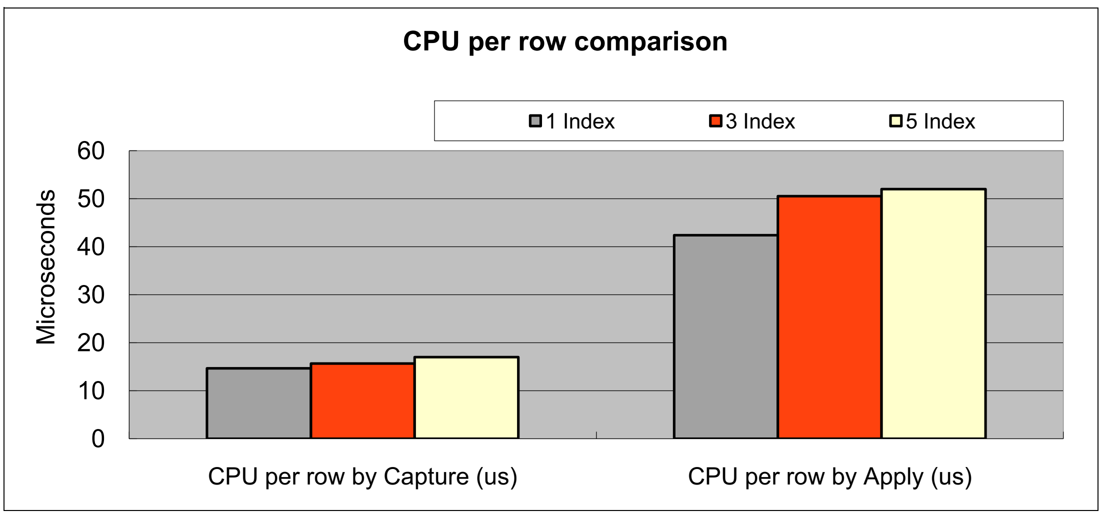


*Figure 3-8 CPU usage by Q Replication for different columns in the index*

Unique indexes introduce more overhead both in DB2 for maintaining the indexes and in Q Apply for inter-transaction dependency analysis. Q Replication does not check for non-unique index dependencies because they do not introduce duplicates. For instance, if you have a unique index, you must play inserts in order. So, check if they depend on each other (inserting each the same value for those columns) and if so, play them in order. If the index is not unique, you do not care and do not check. A best practice is to define the secondary indexes as non-unique, if possible, and limit the number of columns in composite indexes.

# 4

# Tuning Q Replication

Given a mixed workload, how do you configure your installation? We know that transaction size and MQ message size have a particularly significant impact on performance, so how do you optimize the configuration for maximizing them? Reducing DB2, MQ, and Q Replication overhead?

In this chapter, we look at the Q Replication parameters that have potential impact on performance.

This chapter discusses the following topics:

- Column suppression
- Number of Q Apply agents
- TRANS_BATCH_SZ
- MEMORY_LIMIT in Q Capture
- COMMIT_INTERVAL in Q Capture
- MEMORY_LIMIT in Q Apply

# 4.1 Column suppression

With Q Replication, message size has a direct impact on performance. One way to achieve smaller messages is to use the Q Replication column suppression option whenever possible. With column suppression, for SQL updates, only the changed values and the replication key need to be sent to the target. For SQL deletes, only the replication key is sent to the target. See Table 4-1.

*Table 4-1 Column suppression versus no column suppression*

| Test # | No column suppression | Column suppression |
|---|---|---|
| Capture throughput (rows/sec) | 43.3K | 45.1K |
| Message size | 5.8 KB | 3.1 KB |
| Channel throughput (messages/sec) | 2927 | 3039 |
| Channel throughput (MBps) | 17 MB | 9.7 MB |
| Apply throughput (rows/sec) | 43.3K | 45.1K |
| END2END_LATENCY (ms) | 1051 | 1470 |
| QLATENCY (ms) | 43 | 31 |
| Q APPLY_LATENCY (ms) | 441 | 873 |
| Q CAPTURE_LATENCY (ms) | 566 | 565 |
| Capture LPAR MVS BUSY | 30 | 28 |
| Q Apply LPAR MVS BUSY | 83 | 80 |
| Q Capture CPU cost per row (µs) | 14.7 | 13.1 |
| Q Apply CPU cost per row (µs) | 42.4 | 39 |
| Apply DB2 class 2 elapsed time (ms) | 1.95 | 1.895 |

The chart in Figure 4-1 shows the comparison of throughput for column suppression.

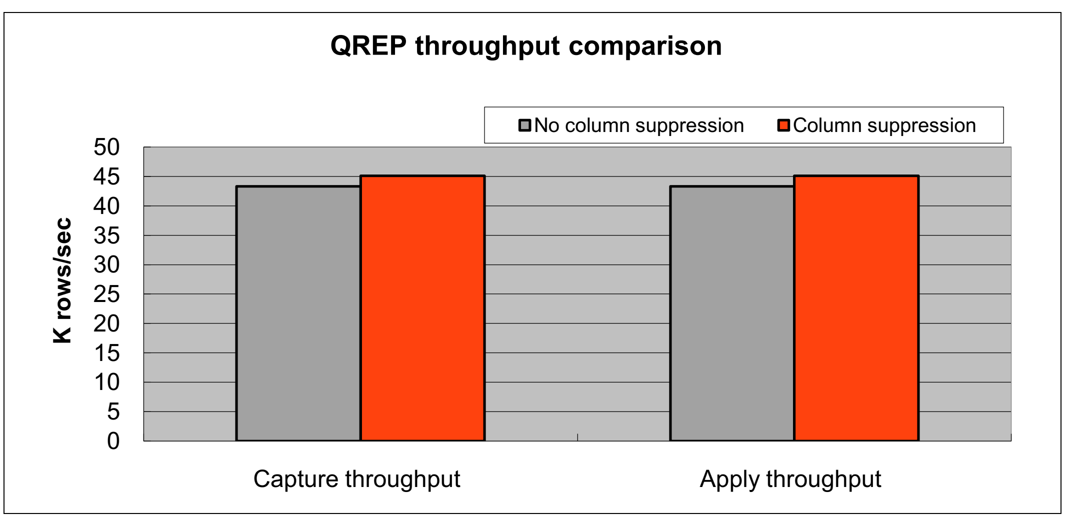



*Figure 4-1 Q Replication throughput comparison for column suppression and no column suppression*

The chart in Figure 4-2 shows the CPU cost for column suppression.

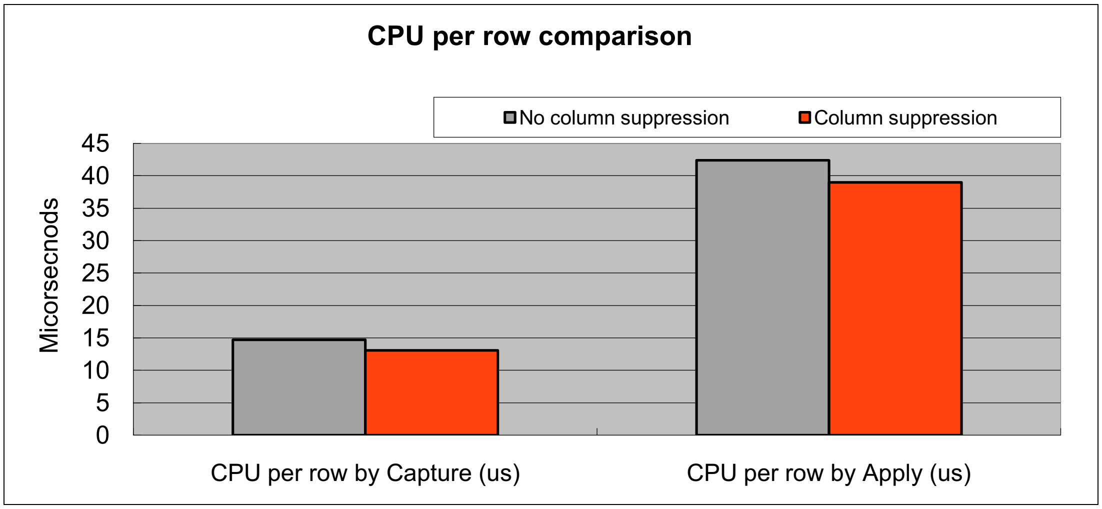



*Figure 4-2 Q Replication performance with column suppression*

The conflict rules determine how much of the data is checked to detect a conflict and the types of conflicts that are detected. When you choose to have more data checked for conflicts, then the Q Capture program must send more data to the Q Apply program to make that data available to be checked. This transmission might influence performance and network traffic. Column suppression (using conflict_action = 'I') specifies that only changed columns are sent. This is good for performance, especially when you have a large number of columns in the table but just a few columns are changed by the SQL statement. The performance improvement can be significant. On the other hand, using column suppression leads to less conflict detection as the cost of higher performance.

## 4.2 Number of Q Apply agents

The num_apply_agents parameter determines how many agent threads are used by the Q Apply program to take reconstructed transactions from the browser and apply them to target tables. A value higher than 1 allows the Q Apply program to process transactions in parallel. More agents allow more parallelism and throughput, as long as system resources are available and the workload does not introduce serialization or contention.

To demonstrate performance improvement with more apply agents, we increase the CPU capacity on LPAR KA (workload) and KC (Q Apply) from 3 CPs to 9 CPs. So we can drive higher workload and run apply with more agents.

Table 4-2 compares the throughput using 1, 4, 6, 8, 12, and 16 agents. It also lists the maximum throughput for different Apply agents, latency, and other indictors.

*Table 4-2 Performance data of using different number of apply agents*

| Test # | Number of Apply agents | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 4 | 6 | 8 | 12 | 16 |
| Capture throughput (rows/sec) | 6K | 37.3K | 52K | 57.8K | 68.7K | 73K |

| Test # | Number of Apply agents | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 4 | 6 | 8 | 12 | 16 |
| Message size | 5.8 KB | 5.8 KB | 5.8 KB | 5.8 KB | 5.8 KB | 5.8 KB |
| Channel throughput msgs/sec | 410 | 2524 | 3470 | 3900 | 4629 | 4876 |
| Channel throughput (MBps) | 2.4 MB | 14.7 MB | 21 MB | 22.7 MB | 27 MB | 28 MB |
| Apply throughput (rows/sec) | 6K | 37.2K | 52K | 57.7K | 68.7K | 72.3K |
| END2END_LATENCY (ms) | 635 | 691 | 750 | 701 | 827 | 2696 |
| QLATENCY (ms) | 67 | 40 | 42 | 44 | 168 | 2050 |
| APPLY_LATENCY (ms) | 31 | 88 | 130 | 66 | 40 | 15 |
| CAPTURE_LATENCY (ms) | 536 | 562 | 578 | 590 | 616 | 629 |
| Q Capture LPAR MVS BUSY | 6 | 25 | 35 | 38 | 46 | 48 |
| Q Apply LPAR MVS BUSY | 5 | 27 | 40 | 44 | 52 | 56 |
| Q Capture CPU cost per row (µs) | 15.3 | 14.2 | 14.4 | 14.2 | 14.4 | 14 |
| Q Apply CPU cost per row (µs) | 38 | 45.6 | 48.9 | 48.6 | 49 | 50 |
| Q Apply DB2 class 2 elapsed time (ms) | 1.34 | 1.48 | 1.59 | 1.89 | 2.3 | 2.3 |

The chart in Figure 4-3 on page 29 shows the throughput improvements with more apply agents.

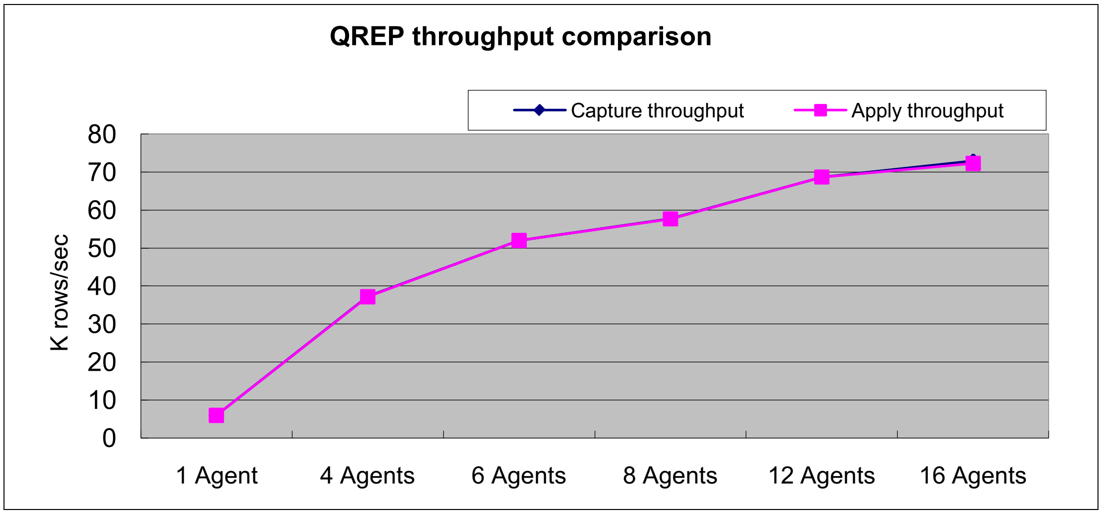



*Figure 4-3 Q Replication throughput with different number of apply agents*

The chart in Figure 4-4 shows the CPU usage per row by Q Replication.

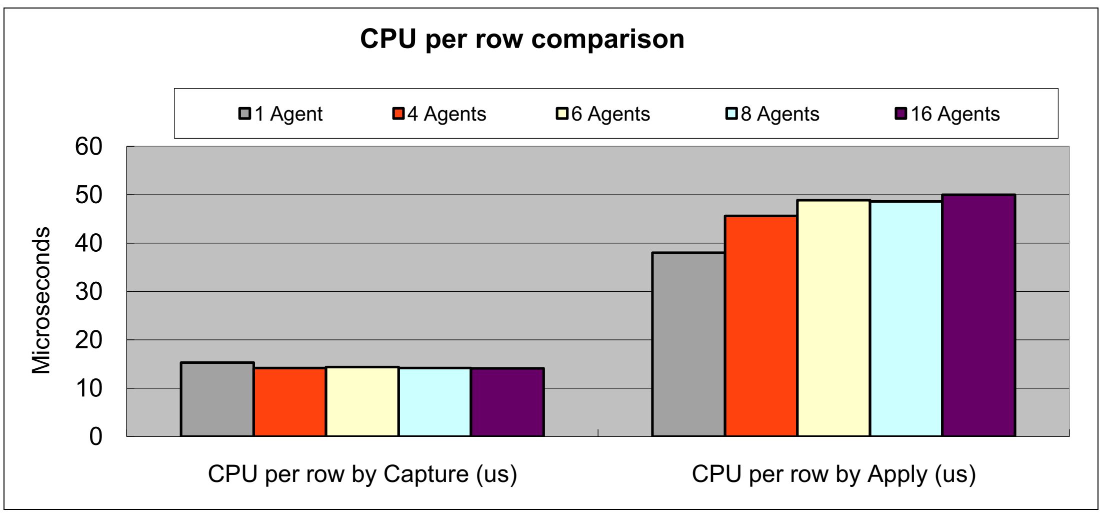



*Figure 4-4 CPU usage by Q Replication of using different numbers of agents*

With 16 agents, Apply can run much faster, which makes MQ the bottleneck. The chart in Figure 4-5 on page 30 shows that all of the latency is in MQ when increasing the workload.

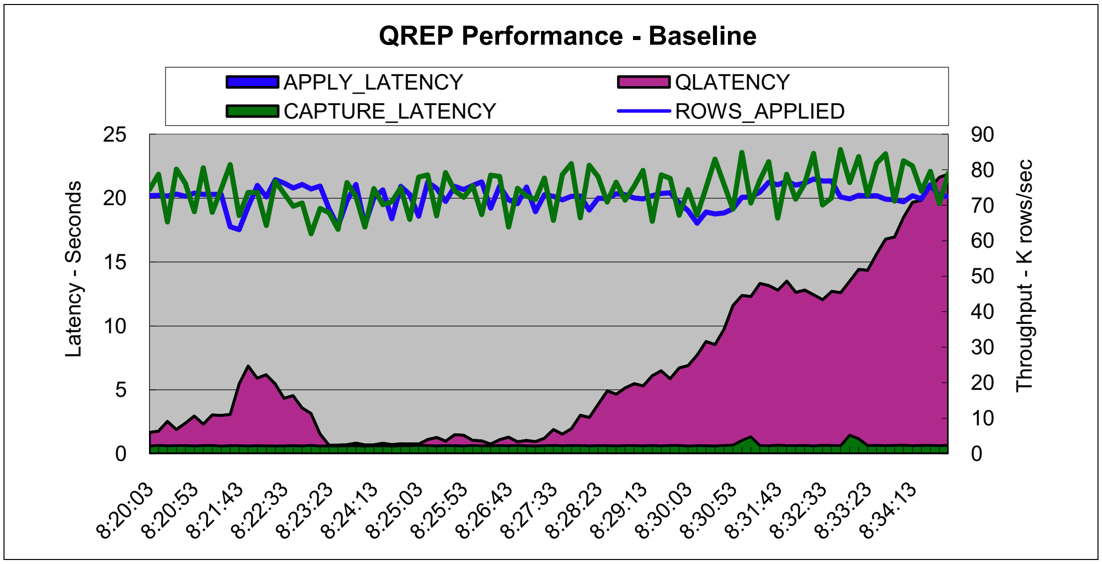



*Figure 4-5 Q Replication performance with 16 agents*

Messages are built in the transmission queue on the source side. MQ channel cannot transmit messages fast enough. The chart in Figure 4-6 shows the queue depth movement and the message in/out rate in this queue.

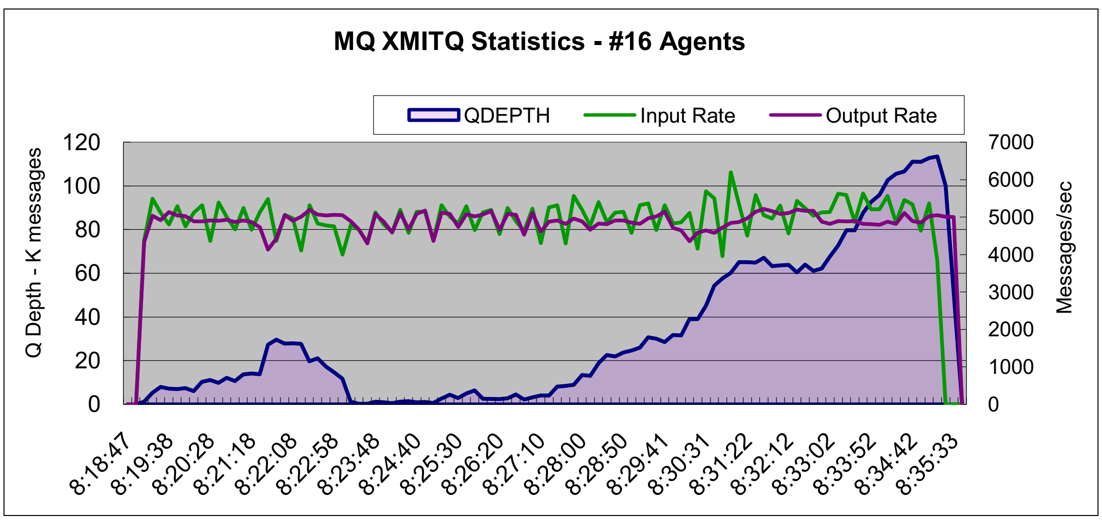



*Figure 4-6 MQ statistic data with 16 agents*

From the testing data, we can see that increasing the apply agents improves Q Apply throughput, but it costs slightly more in CPU. The tuning goal for this parameter is to use the smallest number of agents to handle the workload to minimize the number of active DB2 connections and associated memory requirements for the apply program. That is because each agent eventually has to prepare and cache DB2 SQL statements along with associated data structures for each table that is replicated. Under certain workloads, the memory requirements for Q Apply are proportional to the number of agents' times the number of tables replicated.

## 4.3 TRANS_BATCH_SZ

The TRANS_BATCH_SZ parameter determines the number of source database transactions that Q Capture groups together in a MQ message. Grouping small transactions into a single MQ message can improve MQ transmission throughput for small messages, but it might increase the possibility of lock contention on the apply side due to grouping. Applying concurrently depends on application behavior.

In this scenario, we compared the throughput using TRANS_BATCH_SZ 1, 3, and 5.

Table 4-3 lists the maximum throughput of using different TRANS_BATCH_SZ, latency, and other indicators.

*Table 4-3 Performance data of using different TRANS_BATCH_SZ*

| **Test #** | **TRANS_BATCH_SZ** | | |
|---|---|---|---|
| | **1** | **3** | **5** |
| Capture throughput (rows/sec) | 20.9K | 43.3K | 54.9K |
| Message size | 2.2 KB | 5.8 KB | 9.3 KB |
| Channel throughput (messages/sec) | 4216 | 2927 | 2238 |
| Channel throughput (MBps) | 9.8 MB | 17 MB | 20.7 MB |
| Apply throughput (rows/sec) | 20.9K | 43.3K | 54.9K |
| END2END_LATENCY (ms) | 621 | 1051 | 888 |
| QLATENCY (ms) | 47 | 43 | 52 |
| APPLY_LATENCY (ms) | 28 | 441 | 259 |
| CAPTURE_LATENCY (ms) | 545 | 566 | 577 |
| Q Capture LPAR MVS BUSY | 21 | 30 | 33 |
| Q Apply LPAR MVS BUSY | 71 | 83 | 89 |
| Q Capture CPU cost per row (µs) | 17.2 | 14.7 | 13.6 |
| Q Apply CPU cost per row (µs) | 64 | 42.4 | 37.1 |
| Q Apply DB2 class 2 elapsed time (ms) | 1.23 | 1.95 | 2.48 |
| Throughput improvement | | 107% | 27% |

The graph in Figure 4-7 on page 32 shows throughput improvement with larger TRANS_BATCH_SZ.

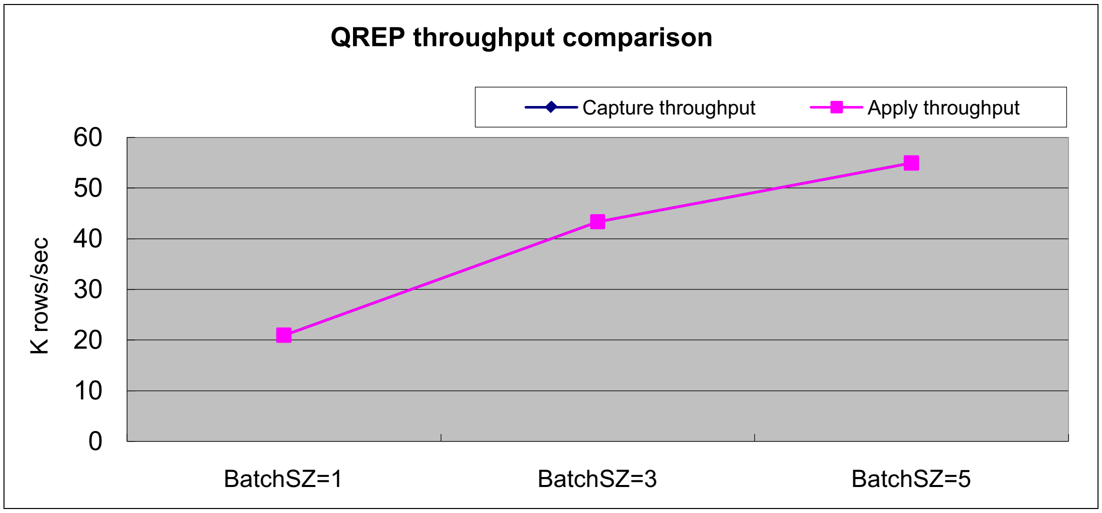


*Figure 4-7 Q Replication throughput with different TRANS_BATCH_SZ*

The chart in Figure 4-8 shows the CPU usage per row by Q Replication.

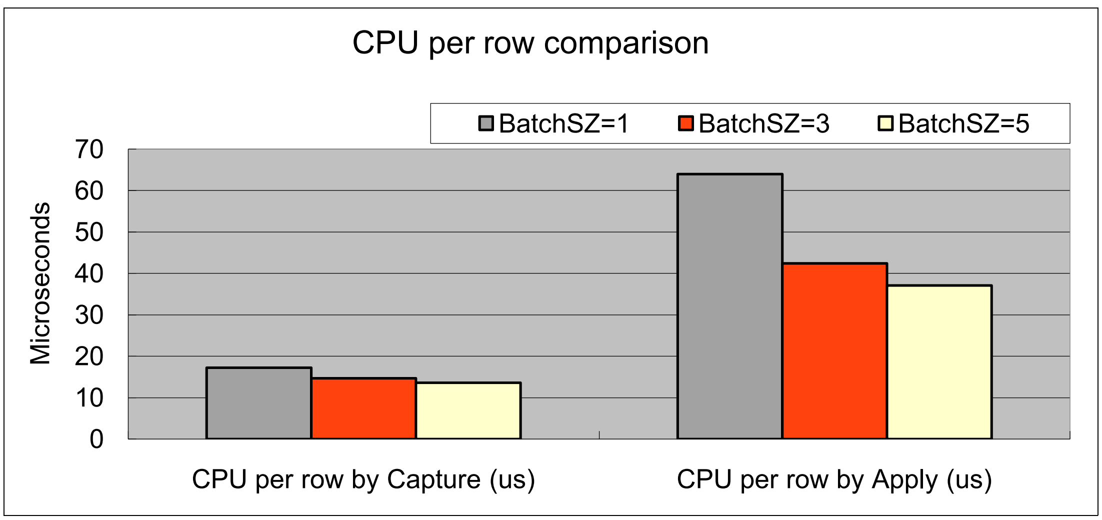


*Figure 4-8 CPU usage by Q Replication for different TRANS_BATCH_SZ*

Using TRANS_BATCH_SZ can improve throughput significantly. It reduces processing in Capture, MQ, and Apply because the message size is larger. We suggest using TRANS_BATCH_SZ as long as no deadlocks or lock timeouts occur at target tables as a result. The nature of our workload, which was modeled upon a real customer workload, did not introduce any contention at the target, so it was a worthwhile tuning choice.

MQ queue managers' logger task uses media manager to perform its I/O, which allows the queue manager to write up to 128 4 KB pages at any time. This suggests that an optimal message size of just under 512 KB (allowing for message headers) can achieve the best logging performance. When we add in channels, the dispatcher process takes the message to be sent and breaks it up into 32 KB chunks of data which are then sent over the channel. Based on this, we might predict the best throughput on a channel to be achieved with 32 KB messages. A sweet spot for replication performance is when the average MQ message size is around 32 KB to 256 KB.

The ability to tune TRANS_BATCH_SZ for meeting this size range can yield the best results.

## 4.4 MEMORY_LIMIT in Q Capture

The MEMORY_LIMIT parameter specifies the amount of memory that a Q Capture program can use to build transactions in memory. In this scenario, we compare the throughput using MEMORY_LIMIT=200 MB, 500 MB, and 800 MB. See Table 4-4.

*Table 4-4 Performance data of using different MEMORY_LIMIT*

| Test # | MEMORY_LIMIT | | |
|---|---|---|---|
| | **200 MB** | **500 MB** | **800 MB** |
| Capture throughput (rows/sec) | 53K | 52K | 53.5K |
| Message size | 5.8 KB | 5.8 KB | 5.8 KB |
| Channel throughput (messages/sec) | 3553 | 3491 | 3622 |
| Channel throughput (MBps) | 20 MB | 20 MB | 21 MB |
| Apply throughput (rows/sec) | 53K | 52K | 53.5K |
| END2END_LATENCY (ms) | 694 | 750 | 691 |
| QLATENCY (ms) | 43 | 42 | 46 |
| APPLY_LATENCY (ms) | 72 | 130 | 66 |
| CAPTURE_LATENCY (ms) | 579 | 578 | 580 |
| Q Capture LPAR MVS BUSY | 35 | 35 | 35 |
| Q Apply LPAR MVS BUSY | 40 | 40 | 40 |
| Q Apply DB2 class 2 elapsed time (ms) | 1.57 | 1.59 | 1.56 |

The graph in Figure 4-9 shows throughput comparison with different MEMORY_LIMIT.

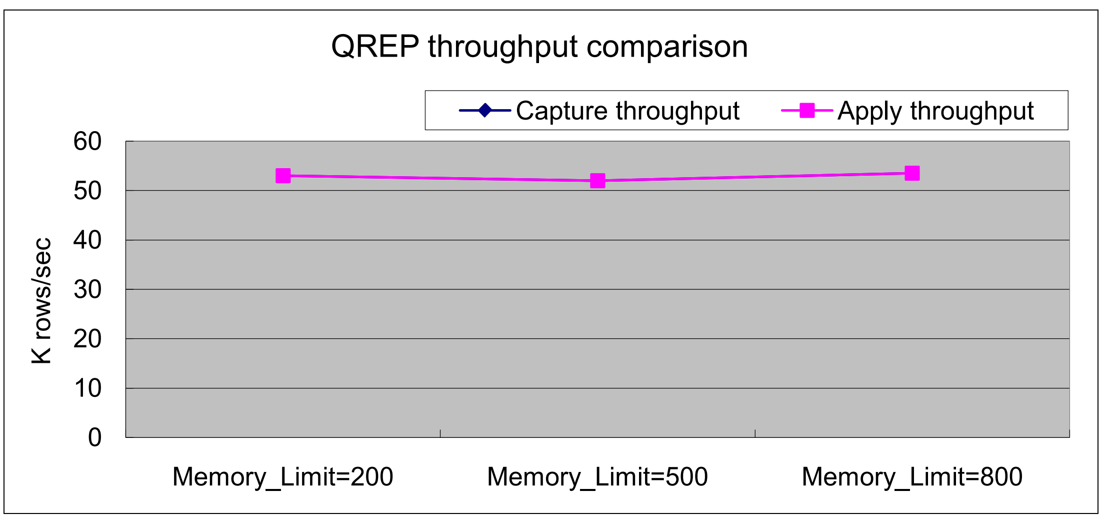



*Figure 4-9 Q Replication throughput with different MEMORY_LIMIT*

The graph in Figure 4-10 on page 34 shows the memory usage for Q Capture.

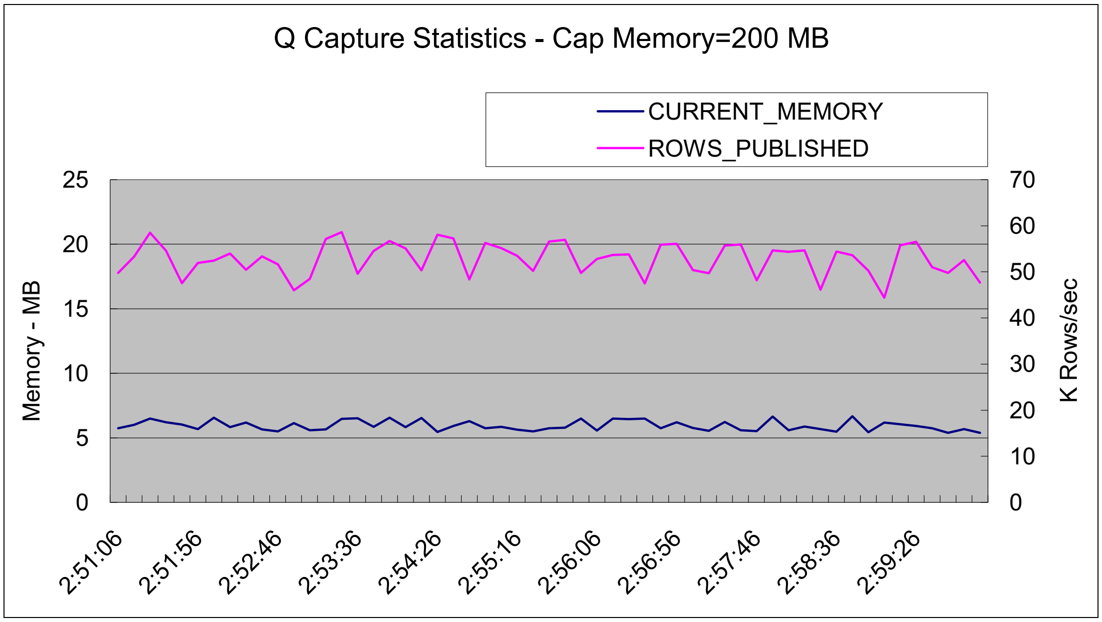



*Figure 4-10 Q Capture memory usage for MEMORY_LIMIT=200 MB*

The MEMORY_LIMIT parameter specifies the amount of memory that a Q Capture program can use to build DB2 transactions in memory. Make it large enough to avoid a *monster* (a trans that does not fit in memory. Extra memory is not detrimental. Q Replication only uses what it needs.

## 4.5 COMMIT_INTERVAL in Q Capture

The COMMIT_INTERVAL parameter specifies how often, in milliseconds, a Q Capture program commits transactions to MQ. In this scenario, we compare the throughput using COMMIT_INTERVAL=200 ms, 500 ms, and 800 ms. See Table 4-5.

*Table 4-5 Performance data when using different COMMIT_INTERVAL values*

| Test # | COMMIT_INTERVAL | | |
|---|---|---|---|
| | **200 ms** | **500 ms** | **800 ms** |
| Capture throughput (rows/sec) | 52K | 52K | 52.8K |
| Message size | 5.8 KB | 5.8 KB | 5.8 KB |
| Message per commit | 43 | 43 | 43 |
| Channel throughput (messages/sec) | 3506 | 3491 | 3564 |
| Channel throughput (MBps) | 20.4 MB | 20 MB | 20.7 MB |
| Apply throughput (rows/sec) | 52K | 52K | 52.8K |
| END2END_LATENCY (ms) | 670 | 750 | 686 |
| QLATENCY (ms) | 41 | 42 | 41 |
| APPLY_LATENCY (ms) | 53 | 130 | 66 |
| CAPTURE_LATENCY (ms) | 576 | 578 | 579 |

| Test # | COMMIT_INTERVAL | | |
|---|---|---|---|
| | 200 ms | 500 ms | 800 ms |
| Capture LPAR MVS BUSY | 35 | 35 | 35 |
| Apply LPAR MVS BUSY | 40 | 40 | 40 |
| Apply DB2 class 2 elapsed time (ms) | 1.58 | 1.59 | 1.55 |

The graph in Figure 4-11 shows throughput comparison with different COMMIT_INTERVAL.

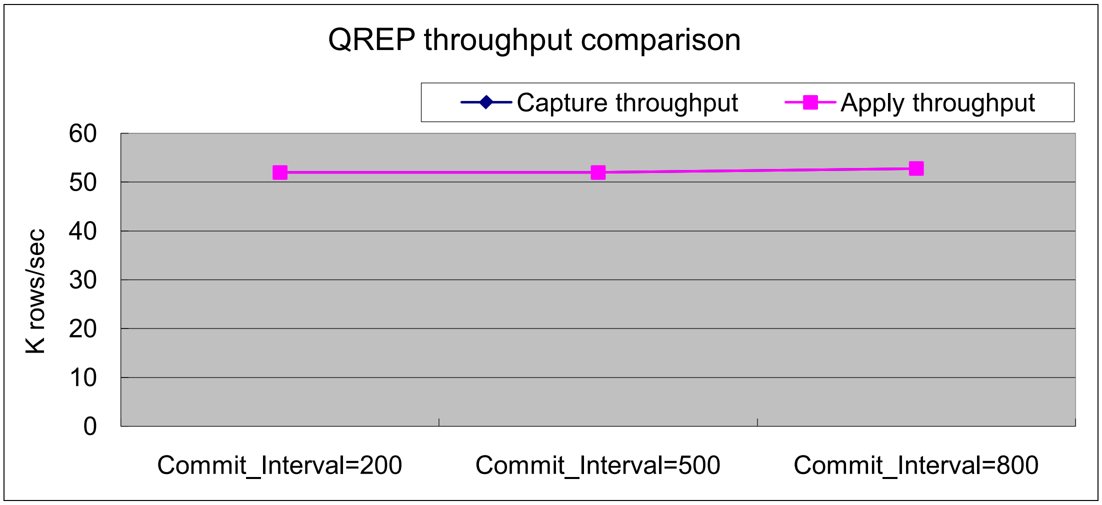



*Figure 4-11 Q Replication throughput with different COMMIT_INTERVAL*

COMMIT_INTERVAL impacts the way Q Capture works with MQ. Lowering this value is not likely to improve throughput, but potentially improves latency. It also flushes messages from memory more often preventing the Q Capture program from spilling transactions to disk.

## 4.6 MEMORY_LIMIT in Q Apply

The MEMORY_LIMIT parameter determines the amount of memory that a Q Apply program can use as a buffer to process transactions from a receive queue. In this scenario, we compare the throughput using Apply MEMORY_LIMIT=100 MB and 200 MB. See Table 4-6.

*Table 4-6 Performance data of using different MEMORY_LIMIT values in Q Apply*

| Test # | MEMORY_LIMIT | |
|---|---|---|
| | 100 MB | 200 MB |
| Capture throughput (rows/sec) | 51.6K | 52K |
| Message size | 5.8 KB | 5.8 KB |
| Channel throughput (messages/sec) | 3470 | 3491 |
| Channel throughput (MBps) | 20 MB | 20 MB |
| Apply throughput (rows/sec) | 51.5K | 52 K |
| END2END_LATENCY (ms) | 671 | 750 |

| Test # | MEMORY_LIMIT | |
|---|---|---|
| | 100 MB | 200 MB |
| QLATENCY (ms) | 41 | 42 |
| APPLY_LATENCY (ms) | 53 | 130 |
| CAPTURE_LATENCY (ms) | 577 | 578 |
| Q Capture LPAR MVS BUSY | 34 | 35 |
| Q Apply LPAR MVS BUSY | 38 | 40 |
| Q Apply DB2 class 2 elapsed time (ms) | 1.60 | 1.59 |

The chart in Figure 4-12 shows the comparison of throughput for the MEMORY_LIMIT values.

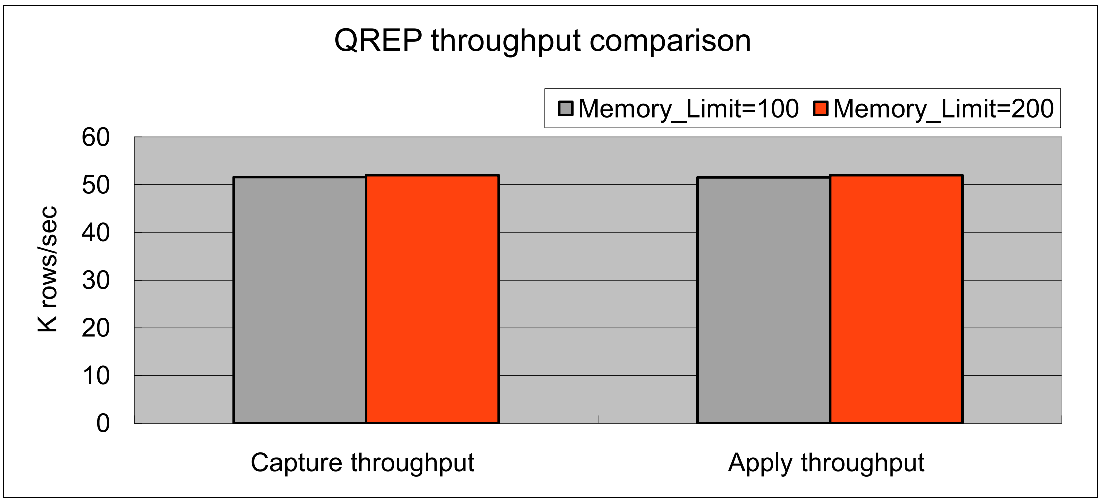



*Figure 4-12 Q Replication throughput comparison for MEMORY_LIMIT in Q Apply*

Apply memory is related to the number of subscriptions, number of agents, and workload variance. The Q Apply MEMORY_LIMIT has no impact on performance in this test. It can be tuned accordingly. Larger memory size does not improve performance unless there is an indicator of lack of memory.

5

# Tuning MQ

In this chapter, we look at some MQ parameters that have potential impact on performance.

This chapter discusses the following topics:

- Persistent messages versus non-persistent messages
- Buffer pool read-ahead
- MQ GLOBAL and CHINIT trace on versus off
- Security on versus security off
- Shared queues versus non-shared queues

## 5.1 Persistent messages versus non-persistent messages

Q Replication can write non-persistent messages to all queues. In this case, messages are not logged and cannot be recovered. The drawback of using non-persistent messages is that a failure of MQ can result in the loss of messages. Recovery requires a manual procedure to restart Capture in the past. With the [1]= MAXCMTSEQ= override, Capture starts parameters to resend the lost messages, or to do a full refresh at the target. There are some cases where restart with LSN poses some challenges, for example, if DDL or new subscriptions were activated in the interval during which the messages were lost. For business-critical applications, persistent messages avoid the complications that can arise when recovering lost messages.

Table 5-1 lists the maximum throughput we achieved for persistent message and non-persistent message, also listing latency and other indicators.

*Table 5-1 Persistent message versus non-persistent message*

| Test # | Persistent message | Non-persistent message |
|---|---|---|
| Q Capture throughput (rows/sec) | 43.3K | 45.9K |
| Message size | 5.8 KB | 5.8 KB |
| Channel throughput (messages/sec) | 2927 | 3100 |
| Channel throughput (MBps) | 17 MBps | 18 MBps |
| Q Capture MQ put elapse time (µs) | 43 | 13 |
| Q Capture MQ commit elapsed time (µs) | 951 | 431 |
| Q Capture MQ log rate (MBps) | 40 | 0.8 |
| Q Apply MQ log rate (MBps) | 36.5 | 0 |
| Q Apply MQ get elapse time (µs) | 42 | 10 |
| Q Apply throughput (rows/sec) | 43.3K | 45.9K |
| END2END_LATENCY (ms) | 1051 | 706 |
| QLATENCY (ms) | 43 | 25 |
| APPLY_LATENCY (ms) | 441 | 114 |
| CAPTURE_LATENCY (ms) | 566 | 566 |
| Q Capture LPAR MVS BUSY | 30 | 28 |
| Q Apply LPAR MVS BUSY | 83 | 84 |
| Q Apply DB2 class 2 elapsed time (ms) | 1.95 | 1.819 |

The chart in Figure 5-1 on page 39 shows the comparison of throughput for persistent and non-persistent messages.

[1] Log sequence number

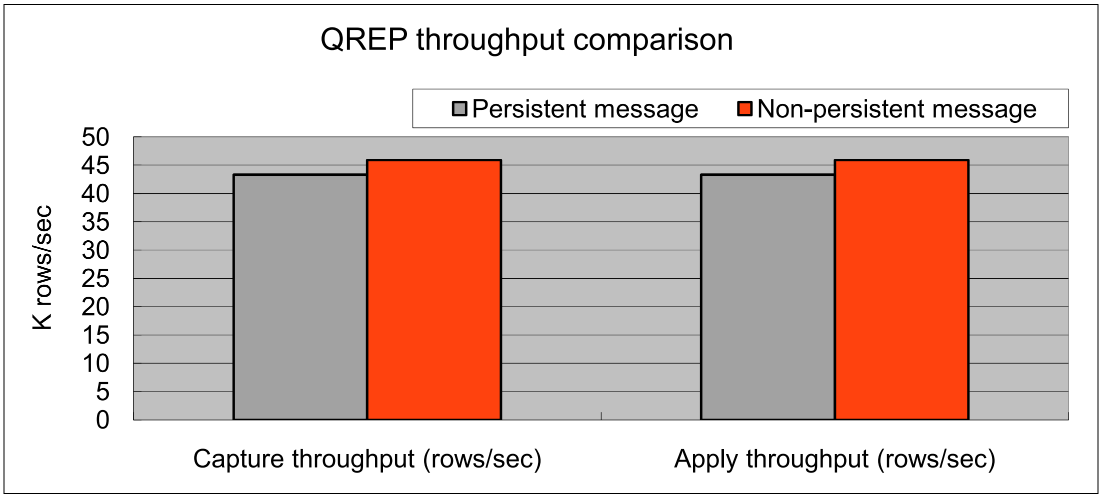



*Figure 5-1 Q Replication throughput comparison for persistent and non-persistent messages*

The chart in Figure 5-2 shows the performance behavior during the test. The throughput is stable and end-to-end latency is low.

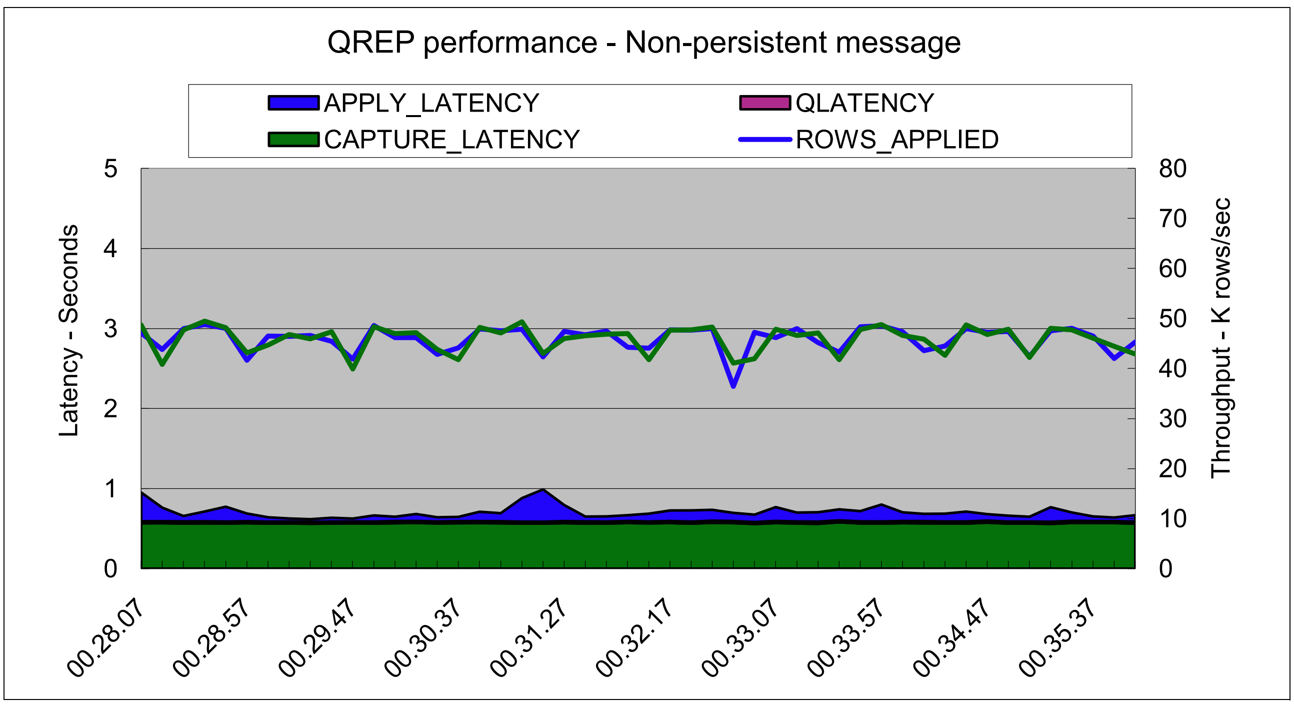



*Figure 5-2 Q Replication performance with non-persistent messages*

Using non-persistent messages only slightly improves Q Replication throughput and end-to-end latency because the MQ buffer pool hit ratio is 100% in our tests. Messages are accessed from memory rather than being accessed from the page set.

There can be a more noticeable improvement for non-persistent measurement if messages need to be accessed from a page set. In Active/Active configurations, we suggest using persistent message to recover the MQ messages in a disaster case.

## 5.2 Buffer pool read-ahead

In this test, we enlarge the MQ page size for the receive queue to 20 GB, so that we can stage a large amount of messages in the receive queue and then start Apply to process the backlog messages. In this way, we can test the efficiency of the read-ahead function to retrieve messages from the page set. We compare the throughput between read-ahead on and off in Figure 5-2 on page 39.

*Table 5-2 MQ read-ahead on versus off*

| Test # | MQ read-ahead off | MQ read-ahead on |
|---|---|---|
| Message size | 5.8 KB | 5.8 KB |
| Q Apply throughput (rows/sec) | 31.5K | 50K |
| Q Apply LPAR MVS BUSY | 61 | 90 |
| Q Apply DB2 class 2 elapsed time (ms) | 1.14 | 1.66 |
| Q Apply MQ Get elapsed time (µs) | 372 | 63 |
| Initial receive queue depth | 2129502 | 2058704 |

The chart in Figure 5-3 compares the average Apply throughput. There is 59% improvement when read-ahead is enabled.

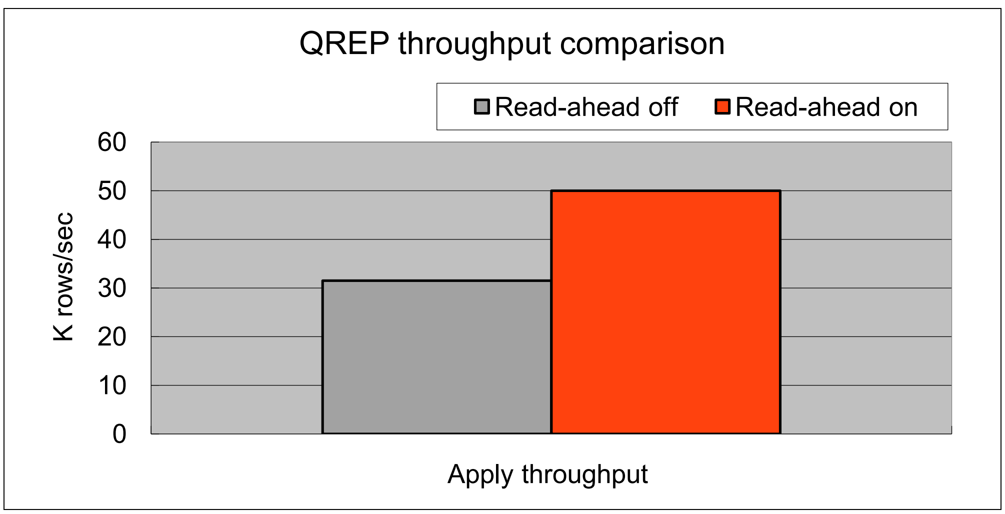



*Figure 5-3 Q Replication throughput comparison for read-ahead on/off*

The chart in Figure 5-4 on page 41 compares the elapsed time in MQ when Apply browses a message from the receive queue.

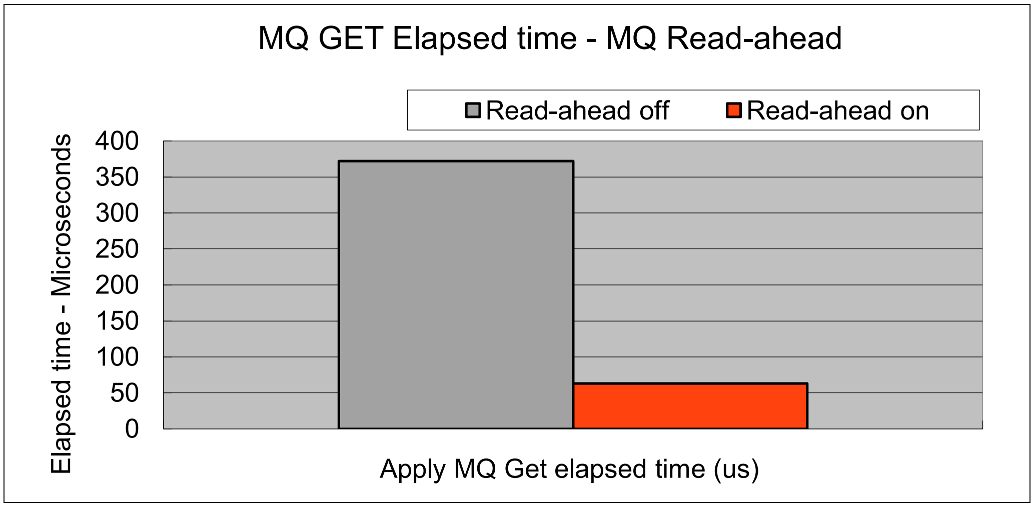



*Figure 5-4 MQ get elapsed time comparison for read-ahead on/off*

The apply throughput is more stable when read-ahead is enabled.

The chart in Figure 5-5 shows the performance numbers for the configuration with read-ahead on.

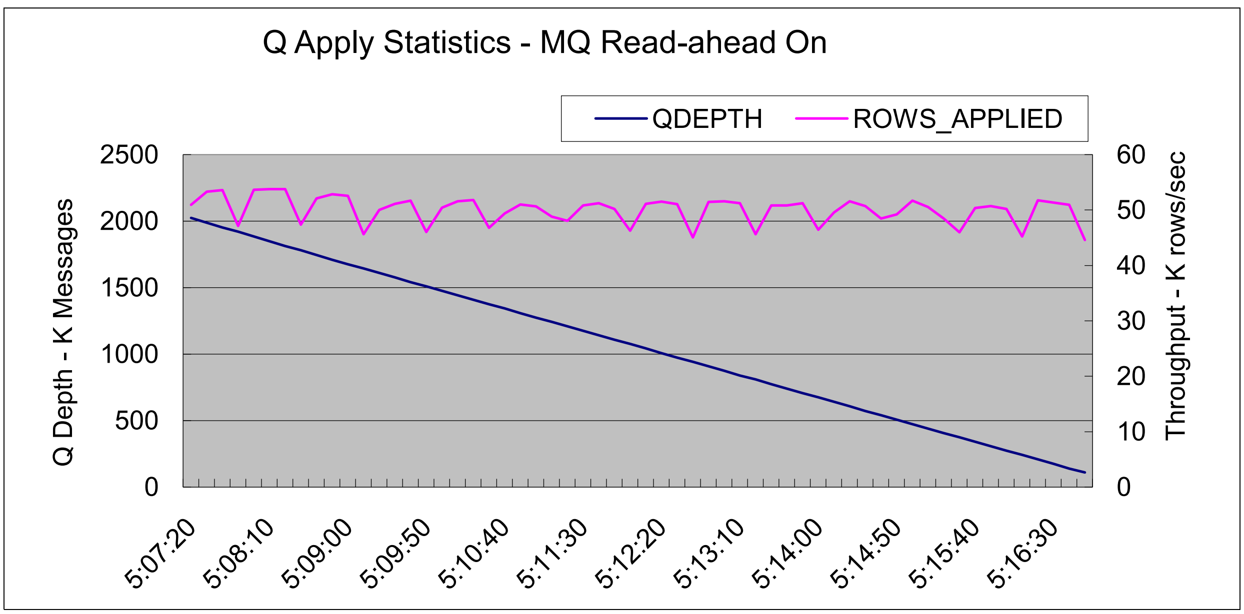



*Figure 5-5 Q Replication performance with MQ read-ahead on*

The chart in Figure 5-6 on page 42 shows the performance numbers for the configuration with read-ahead off.

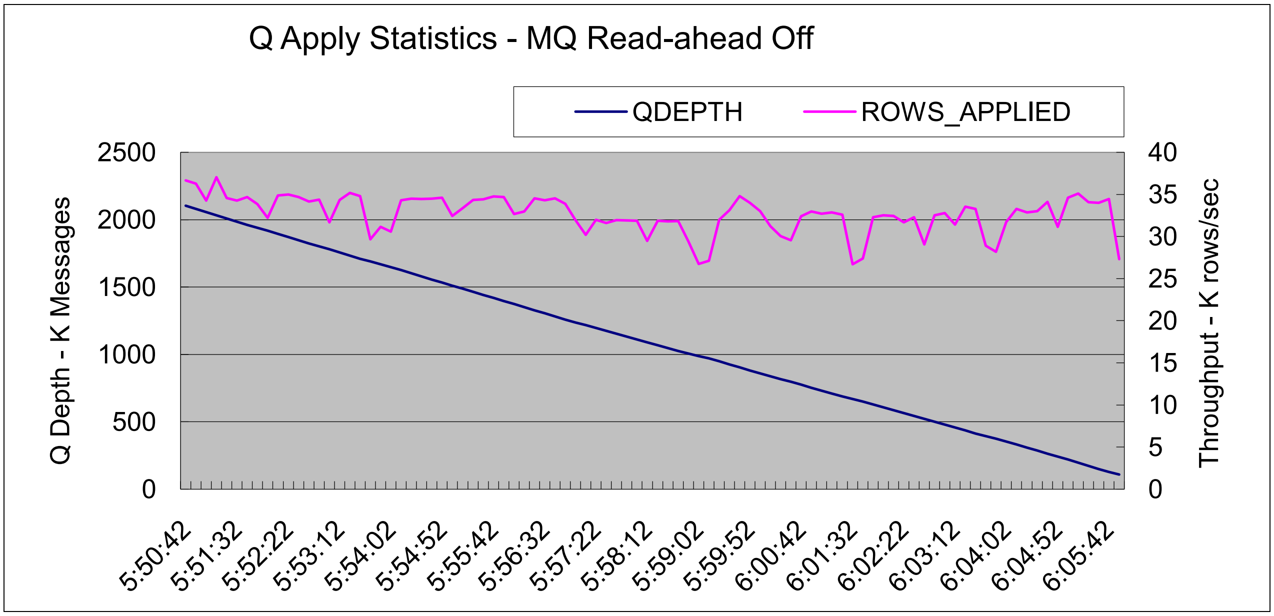



*Figure 5-6 Q Replication performance with MQ read-ahead off*

When the number of messages overruns the buffer pool allocated for the queue, messages are spilled to disk and must then be retrieved from disk. With the read-ahead enhancement, when Q Apply requests a message, the message is in memory. In addition to greatly improving throughput in these situations, the enhancement lowers overall replication latency. This new MQ feature also benefits performance at the source transmission queue.

## 5.3 MQ GLOBAL and CHINIT trace on versus off

MQ trace can be used to determine problems and gather performance data. In all previous tests, we started the accounting trace and statistic trace to collect performance data.

Table 5-3 compares the impact of the trace being set on versus being set off.

*Table 5-3 MQ GLOBAL and CHINIT[2] trace on versus off*

| Test # | MQ internal trace on | MQ internal trace off |
|---|---|---|
| Capture throughput (rows/sec) | 42.2K | 43.3K |
| Message size | 5.8 KB | 5.8 KB |
| Channel throughput (Messages/sec) | 2855 | 2927 |
| Channel throughput (MBps) | 16.6 MB | 17 MB |
| Q Apply throughput (rows/sec) | 42.1K | 43.3K |
| END2END_LATENCY (ms) | 833 | 1051 |
| QLATENCY (ms) | 47 | 43 |
| APPLY_LATENCY (ms) | 222 | 441 |
| CAPTURE_LATENCY (ms) | 564 | 566 |

[2] Channel Initiator

| Test # | MQ internal trace on | MQ internal trace off |
|---|---|---|
| Q Capture LPAR MVS BUSY | 34 | 30 |
| Q Apply LPAR MVS BUSY | 89 | 83 |
| Q Apply DB2 class 2 elapsed time (ms) | 1.99 | 1.95 |
| Q Capture MQ Put elapse time (µs) | 84 | 43 |
| Q Apply MQ Get elapsed time (µs) | 88 | 42 |

The chart in Figure 5-7 shows the comparison of throughput.

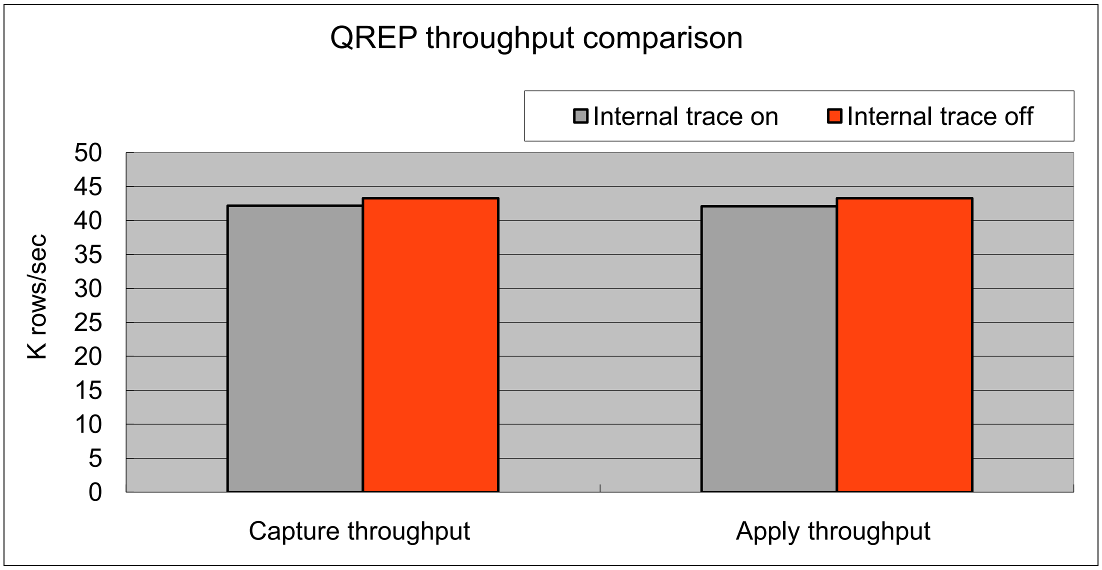



*Figure 5-7 Q Replication throughput comparison for GLOBAL/CHINIT trace on/off*

Figure 5-8 shows the elapsed time comparison.

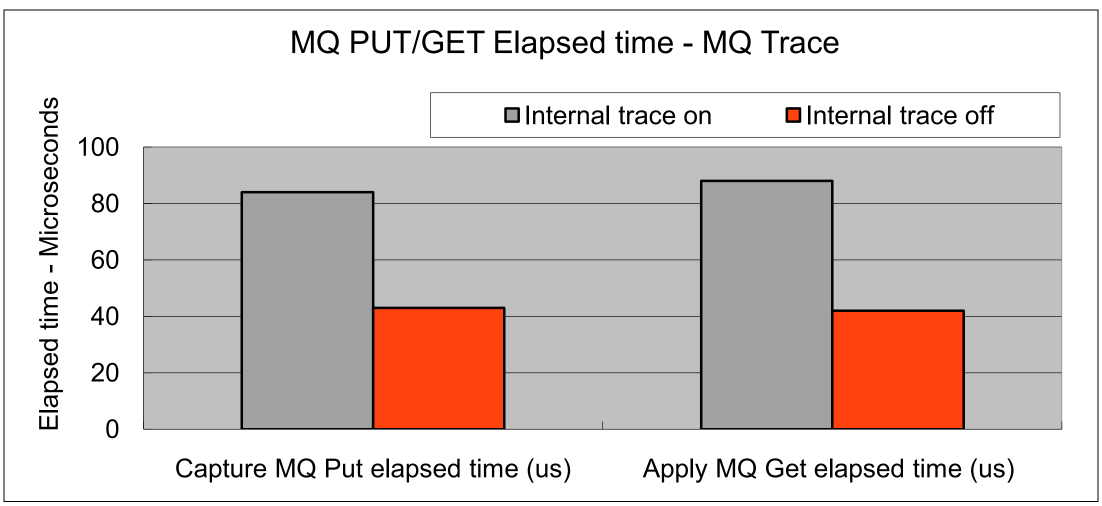



*Figure 5-8 MQ put/get elapsed time comparison between GLOBAL/CHINIT trace on and off*

MQ internal trace has considerable overhead for MQ PUT/GET time. But compared to Q Capture and Q Apply elapsed, the overhead of MQ internal trace is not noticeable; therefore, there is no impact to the overall throughput.

## 5.4 Security on versus security off

In test systems, users tend to turn MQ security off, but in production it must be turned on. In Table 5-4 we compare the performance impact of turning security on.

*Table 5-4 MQ security on versus off*

| Test # | MQ security on | MQ security off |
|---|---|---|
| Capture throughput (rows/sec) | 42.7 K | 42.7 K |
| Message size | 5.8 KB | 5.8 KB |
| Channel throughput (messages/sec) | 2889 | 2878 |
| Channel throughput (MBps) | 16.8 MB | 16.7 MB |
| Apply throughput (rows/sec) | 42.7K | 42.7K |
| END2END_LATENCY (ms) | 662 | 655 |
| QLATENCY (ms) | 48 | 44 |
| APPLY_LATENCY (ms) | 50 | 48 |
| CAPTURE_LATENCY (ms) | 563 | 563 |
| Q Capture LPAR MVS BUSY | 28 | 28 |
| Q Apply LPAR MVS BUSY | 81 | 83 |
| Q Apply DB2 class 2 elapsed time (ms) | 1.86 | 1.89 |
| Q Capture MQ Put elapse time (µs) | 37 | 37 |
| Q Apply MQ Get elapse time (µs) | 38 | 41 |

The chart in Figure 5-9 on page 45 shows the comparison of throughput. The throughput does not change when security is turned on.

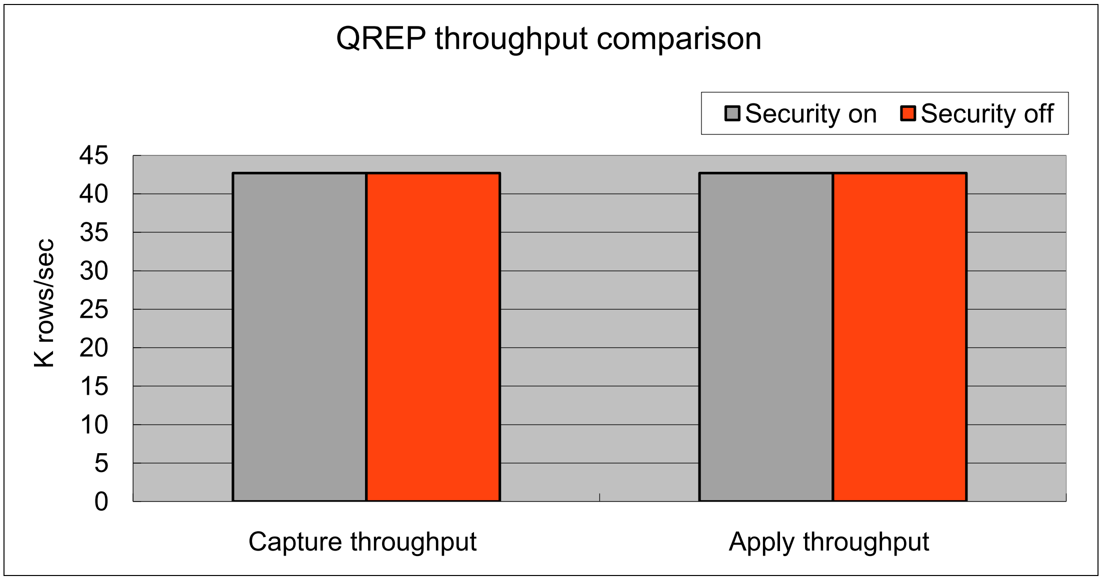



*Figure 5-9 Q Replication throughput comparison for security on/off*

In this test, Figure 5-10, we observed that MQ PUT and GET elapsed times are almost the same as when security is turned off.

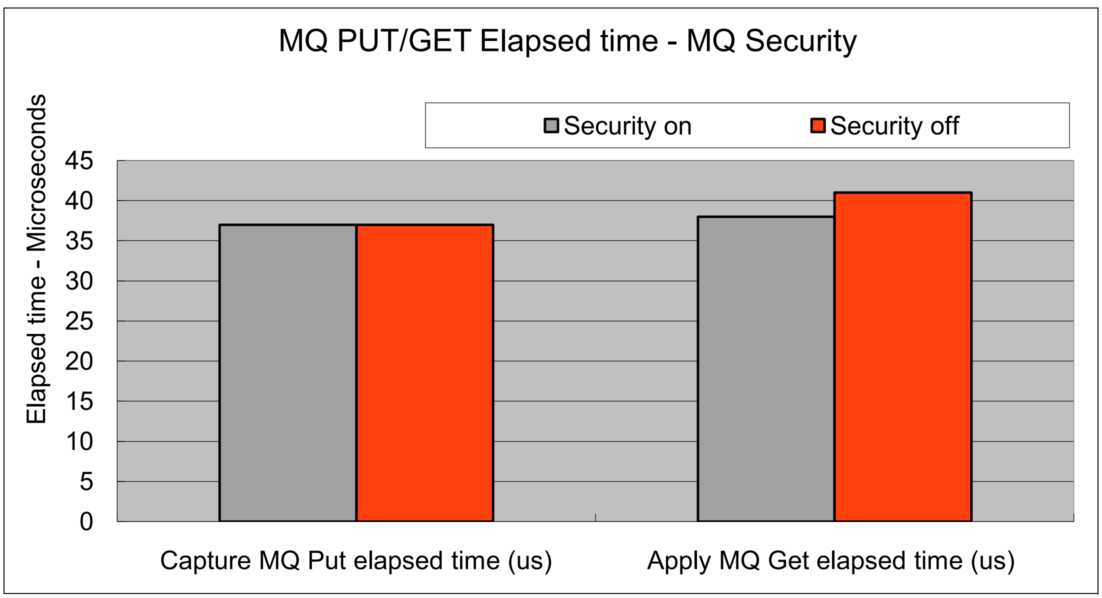



*Figure 5-10 MQ put/get elapsed time comparison between security on/off*

The security cost is most likely incurred at channel start and then at the refresh interval. The remainder of the time it must be cached and does not have much impact to the overall throughput.

## 5.5 Shared queues versus non-shared queues

MQ can be configured to use shared queue, which stores messages in the CF structure instead of the local buffer pool. We compare the performance using private queue versus shared queue.

Figure 5-11 shows the test configuration. We set up two queue-sharing groups and created a MQ Transmission queue and Receive queue as the shared queue. This allows the messages from Q Capture to be loaded in the CF structure.

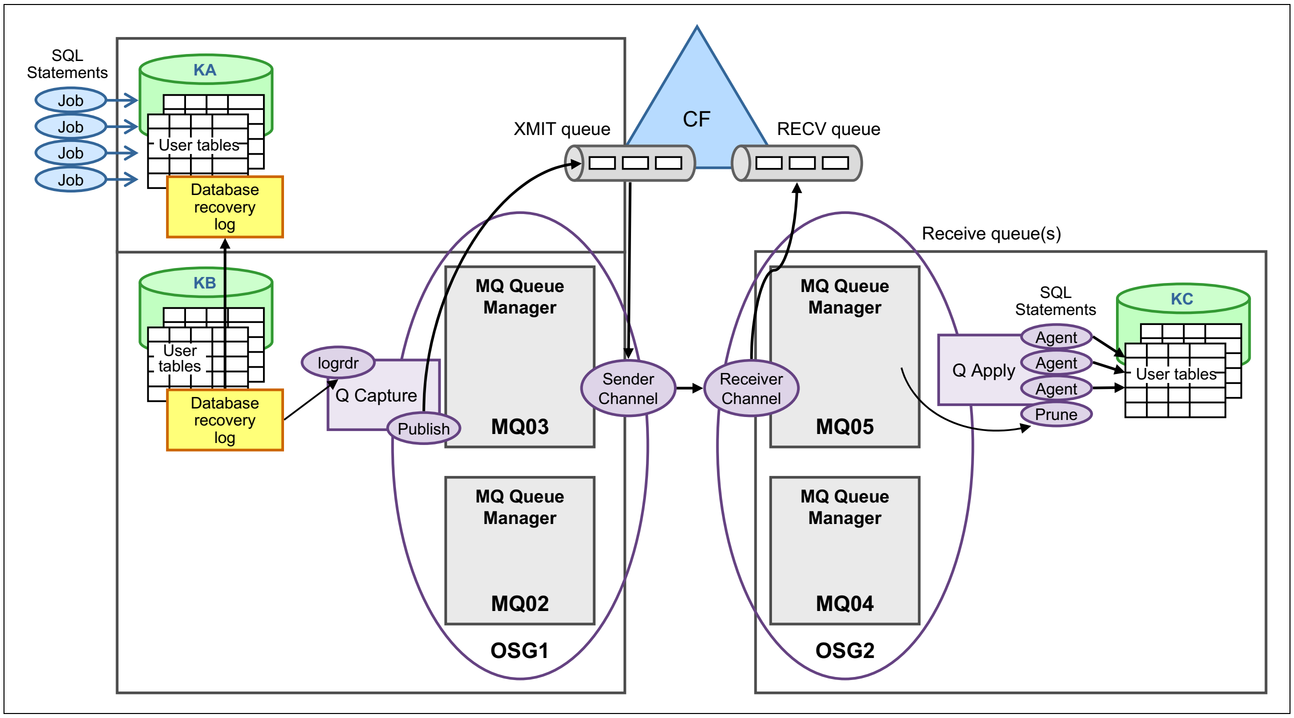



*Figure 5-11 MQ shared queues configuration*

In our test, the internal CF is on the same physical machine of the source and target LPAR, so the network/channel delay of accessing CF is minimum. When all messages are in the buffer pool or CF, there is not much difference for Q Replication throughput between the MQ configuration with private queue or shared queue. See Table 5-5.

*Table 5-5 MQ private queue versus shared queue*

| Test # | Private queue | Shared queue |
|---|---|---|
| Q Capture throughput (rows/sec) | 40.7 K | 40.7 K |
| Message size | 5.8 KB | 5.8 KB |
| Channel throughput (messages/sec) | 2717 | 2716 |
| Channel throughput (MBps) | 15.8 MB | 15.8 MB |
| Q Apply throughput (rows/sec) | 40.4K | 40.4K |
| END2END_LATENCY (ms) | 648 | 813 |
| QLATENCY (ms) | 44 | 66 |
| APPLY_LATENCY (ms) | 43 | 185 |
| CAPTURE_LATENCY (ms) | 561 | 562 |
| Q Capture LPAR MVS BUSY | 27 | 26 |
| Q Apply LPAR MVS BUSY | 80 | 76 |
| Q Apply DB2 class 2 elapsed time (ms) | 1.935 | 1.873 |

In our test, the internal CF is on the same physical machine with the source and target LPAR, so the network/channel delay of accessing CF is minimum. When all messages are in buffer pool or CF, there is not much difference for Q Replication throughput when we configure MQ as a private queue or shared queue.

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

## IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.


- *Smarter Business Dynamic Information with IBM InfoSphere Data Replication CDC,* SG24-7941
- *WebSphere MQ Primer*, REDP-0021-01


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

## Online resources

These websites are also relevant as further information sources:

- InfoSphere Data Replication

  http://www.ibm.com/software/data/infosphere/data-replication/
- IBM Software Information Management InfoSphere Platform: InfoSphere Data Replication for DB2 for z/OS

  http://www.ibm.com/software/data/infosphere/data-replication-db2-z/
- IBM InfoSphere Data Replication version 10.1.3 Information Center

  http://publib.boulder.ibm.com/infocenter/iidr/v10r1m2/index.jsp
- IBM WebSphere MQ version 7.1 Information Center

  http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/index.jsp
- MP16: WebSphere MQ for z/OS - Capacity planning & tuning

  http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg24007421&loc=en_US&cs=utf-8&lang=en
- Q Replication recovery: Advice for Q Replication system management

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0709hamel/index.html

# Help from IBM

IBM Support and downloads

**ibm.com**/support

IBM Global Services

**ibm.com**/services

# Index



# InfoSphere Data Replication for DB2 for z/OS and WebSphere Message Queue for z/OS

## Performance Lessons

Redpaper™

**Q Replication for DB2 for z/OS and WebSphere MQ performance study**

**Performance impact by workload and table characteristics**

**Performance impact by Q Replication and MQ configurations**

Understanding the impact of workload and database characteristics on the performance of both DB2, MQ, and the replication process is useful for achieving optimal performance.
Although existing applications cannot generally be modified, this knowledge is essential for properly tuning MQ and Q Replication and for developing best practices for future application development and database design. It also helps with estimating performance objectives that take these considerations into account.

Performance metrics, such as rows per second, are useful but imperfect. How large is a row? It is intuitively, and correctly, obvious that replicating small DB2 rows, such as 100 bytes long, takes fewer resources and is more efficient than replicating DB2 rows that are tens of thousand bytes long. Larger rows create more work in each component of the replication process. The more bytes there are to read from the DB2 log, makes more bytes to transmit over the network and to update in DB2 at the target.

Now, how complex is the table definition? Does DB2 have to maintain several unique indexes each time a row is changed in that table? The same argument applies to transaction size: committing each row change to DB2 as opposed to committing, say, every 500 rows also means more work in each component along the replication process.

This Redpaper reports results and lessons learned from performance testing at the IBM laboratories, and it provides configuration and tuning recommendations for DB2, Q Replication, and MQ. The application workload and database characteristics studied include transaction size, table schema complexity, and DB2 data type.

**INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

**BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**
**ibm.com**/redbooks

REDP-4947-00