IBM

IBM System z Strengths and Values

- Technical presentation of System z hardware and z/OS
- Enterprise-wide roles for the System z platform
- Cost of computing considerations

Philippe Comte
Andrea Corona
James Guilianelli
Douglas Lin
Werner Meiner
Michel Plouin
Marita Prassolo
Kristine Seigworth
Eran Yona
Linfeng Yu

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM System z Strengths and Values

May 2008

SG24-7333-01

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

First Edition (May 2008)

This edition applies to the IBM System z platform and IBM z/OS V1.8.

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 give 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites 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.

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:

Redbooks (logo) ®
AIX®
CICS®
CUA®
DB2 Connect™
DB2 Universal Database™
DB2®
DFSMS™
DFSMSdfp™
DFSMSdss™
DFSMShsm™
DFSMSrmm™
Distributed Relational Database Architecture™
Domino®
DRDA®
DS8000™
Encina®
ES/9000®
ESCON®
eServer™
FICON®
FlashCopy®
GDPS®
Geographically Dispersed Parallel Sysplex™
HiperSockets™
HyperSwap™
IBM®
IMS™
Informix®
Language Environment®
Lotus®
MVS™
OMEGAMON®
OS/390®
Parallel Sysplex®
PowerPC®
PR/SM™
Processor Resource/Systems Manager™
RACF®
Rational®
Redbooks®
RMF™
S/360™
S/370™
S/390®
Sysplex Timer®
System i™
System p™
System Storage™
System x™
System z™
System z9®
System/360™
System/370™
System/390®
Tivoli Enterprise Console®
Tivoli®
TotalStorage®
VisualAge®
VSE/ESA™
VTAM®
WebSphere®
z/Architecture®
z/OS®
z/VM®
z/VSE™
z9™
zSeries®

The following terms are trademarks of other companies:

mySAP, SAP NetWeaver, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

EJB, Enterprise JavaBeans, J2EE, Java, JavaBeans, JavaServer, JDBC, JDK, JSP, JVM, RSM, Sun, Virtual Storage Manager, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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, other countries, or both.

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

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

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

Preface

This IBM® Redbook describes the strengths and values of the IBM System z™ platform with special focus on IBM z/OS®. Other System z operating systems are described briefly in relation to the values that customers can obtain from them while running on the System z hardware.

The book addresses the technical aspects of the System z environment and focuses on the business value of the System z solution, the overall role that it plays in the business enterprise, and how it offers an effective total cost of computing.

The audience for this book is the IT architect who is not familiar with the System z platform. System z architects will also be interested in this book for understanding the important role that the System z environment plays in the business solution, especially with the deployment of new workloads and to address security, availability, and scalability.

The team that wrote this redbook

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Philippe Comte is a Software Group Senior IT Architect with IBM France with 20 years of experience in IT at IBM. He has participated in technical sales and project assignments in France and other European countries. He is currently in charge of large public sector customers. His areas of expertise include large centralized applications and collaborative and middleware systems. He holds a degree in Business Management from the ESSEC-Paris School of Economics and is the co-author of two Redbooks®: one about Lotus® Domino®, the other about service-oriented architectures (SOA) and Web Services.

Andrea Corona is an IT Specialist working for IBM Systems and Technology Group in Italy with System z technical sales responsibilities. Since he joined IBM three years ago, he has worked in the IBM Global Business Services unit and was involved in customer projects, mainly in the area of Performance Engineering for Web J2EE™ Application and Linux® solutions. His areas of expertise include Application and Systems Performance Engineering, and System z platform and Linux software solutions. He holds a degree in Electronic Engineering from the Universita' di Cagliari, Italy.

James Guilianelli is a writer and editor with IBM Systems and Technology Group in Poughkeepsie, New York. He has worked for IBM for more than 20 years. His areas of interest are MVS™, RACF®, RMF™, and Parallel Sysplex®.

Douglas Lin is a technical Sales Specialist who works with System z clients in New York City. He has worked at IBM for more than seven years and spent most of his career in System z development, where he held various roles, ranging from Fiber Channel and TCP firmware developer to System z9® I/O Integration Manager. Douglas has both a B.S. and Masters in Electrical Engineering from Cornell University.

Werner Meiner is a Senior IT Architect and IBM Certified Professional from Germany. He has 24 years of IT experience with IBM as a consultant in mainframe and database technology, client/server architecture, and IT infrastructure extensively with hardware, software, and services. Currently he works in technical sales as an architect for SAP® infrastructure solutions.

Michel Plouin is a System z Consultant, working since 2000 for DCV Informatique / Groupe ARES, an IBM Business Partner in France. Before that he worked for 32 years for IBM France as a large-system SE and spent several years at the Field Support Center. Out of these 32 years with IBM, he spent more than 10 as an assignee to the ITSO in Poughkeepsie where he published many Redbooks on subjects such as JES2, MVS, Performance, Tuning, and Security. He also spent one and a half years at the IBM Poughkeepsie Laboratory, running the LSPR performance measurements.

Marita Prassolo is an Executive IT Architect working for IBM Systems and Technology Group in Italy with System z technical pre-sales responsibilities. She joined IBM 25 years ago and has been involved in many projects regarding high availability and technology enabling on the mainframe platform. Her expertise includes infrastructure design focused on high availability and performance solutions.

Kristine Seigworth is a System z Architect and High Availability Technical Project Manager. She has more than eight years of experience in software design, development, service, and testing of z/OS. Her area of expertise is memory management. Kristine has a B.S. in Computer Science from Indiana University of Pennsylvania.

Eran Yona is an IT Architect from the Israeli Ministry of Defense with 15 years of experience in IT. He has a BA degree in business from the College of Management in Israel. His areas of expertise include Datacenter management, Datacenter Infrastructure, Disaster Recovery solutions, and virtualization.

Linfeng Yu is a lead architect with Insurance Services Office, Inc. He has extensive experience in developing large-scale, complex enterprise-wide

architectures and cross-platform software development. He has been architecting Java™ enterprise applications on System z and distributed platforms for the past six years. His focus is on SOA now.

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

Paola Bari, Paolo Bruni, Franck Injey
International Technical Support Organization, Poughkeepsie Center

Paul diMarzio, Rich Guski, Carl Parris
IBM Poughkeepsie

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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 Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an e-mail 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

Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition may also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-7333-01
for IBM System z Strengths and Values
as created or updated on May 9, 2008.

May 2008, First Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

Changed information

- Updated to remove obsolete charts.

1

A business view

This chapter describes the business drivers, their impact on IT, and how the System z platform can be a major player for the business solution through:

- Flexibility
- Responsiveness to customers and market needs
- Cost effectiveness

1.1 Business drivers

Businesses today are asking common questions:

- How can we be more flexible and dynamic?
- How can we be more responsive to market needs?
- How can we continue doing business and be more cost effective?

Asking those questions results in looking at some major business drivers that influence the business today:

- Flexibility
- Responsiveness
- Cost effectiveness

Business flexibility

The market today is very different from the market we saw in the past few years. Competition between companies has become very aggressive, markets have become global, and products must be innovated constantly to keep up with customer expectations. In order to compete with each other, businesses need to produce products that will respond to customer needs in a short period of time.

Because of the rapid pace of market changes, companies need to be very flexible with their products and lines of business. In addition to market changes that put strains upon businesses, mergers and acquisitions are common business occurrences that affect the climate today.

Businesses today more than ever need integrated processes. Processes for a single business unit are composed of several smaller processes, some which might be unique to a business unit, some which might be reused from other business units, and some which might be shared with other business units. The organization must be able to analyze specific parts of business processes, and change or outsource these, without interfering with the overall processes.

Responsiveness

Along with flexibility to address change, every company needs to respond quickly to customers' needs, a major goal for any company. The clients today know what they want, the way they want it, and how much they are willing to pay for it. In today's very open and global market, it is easy for customers to switch suppliers when the product offering no longer satisfies them or is not available.

The shifts of customer demands require companies to be able to respond to increases (and, as a matter of fact, decreases) in demand very quickly. A business can serve more customers during a critical business period (such as a

holiday season) without having to worry about reserving the necessary resources for those peak periods.

A customer often needs to deal with a new business direction as quickly as possible. Most business solutions are usually required in a few weeks instead of months or years. The business should know how to reuse its assets and implement solutions quickly to respond to market demands. To build a solution from scratch without reusing assets takes a long time—and that is something businesses cannot afford today.

Cost

The price of the products a company puts on the market must be in line with the competitive strategy. If the company is focused on cost leadership, this is an obvious requirement. But even for companies with differentiation and focus strategies, there is nowadays a constant pressure to be cost effective.

Business processes can be very complex. Businesses today are analyzing and redesigning their processes with the goal to identify and strengthen those processes that provide the company a competitive advantage. Optimizing processes in terms of reuse of resources can help business efficiency and can make businesses more flexible and responsive.

1.2 Impact on IT

In the previous section we discussed the major business drivers companies are facing today. It is no secret that there is a bold connection between the business drivers and the information technology environment. In order to accomplish the business drivers, IT has to deal with infrastructure complexity and reliability, data center management, energy problems, and budgeting costs. The following sections describe these key points.

Simplifying the infrastructure

Modern IT infrastructure consists of multiple hardware architecture, a variety of software operating environments, complex internal and external networking, and applications that must interoperate in order to provide end-to-end service to customers and internal users alike. The flexibility to meet new market demands and to seize opportunities before they are lost to competitors is a must for today's businesses.

In order to support these demands, the IT environment has to support changes on demand to the business needs. The IT systems must be able to communicate and integrate in a quick, flexible way with other IT systems within the company as

well as with systems from partners, suppliers, and consumers in the external environment.

Figure 1-1 shows an architecture design of an IT infrastructure. The number of servers that are needed to manage the environment is huge; there are four different networks to manage: an internal and external IP network, a storage network for the distributed servers (SAN), and the System z connectivity network.

Updating or maintaining the infrastructure can be a difficult job and at times seems impossible. Creating a disaster recovery solution for this environment can also be very complicated and demands continuous updates and maintenance. Backing up and monitoring processes requires careful coordination among all of the components in the infrastructure.

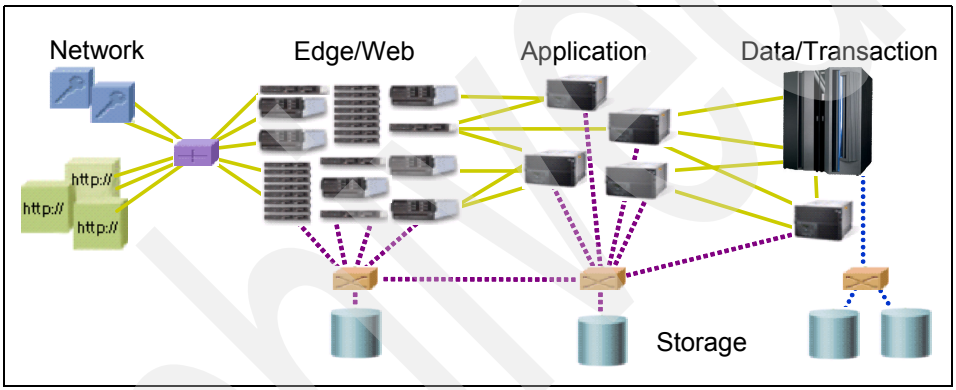


Figure 1-1 Infrastructure complexity

IT complexity impacts business process in that it requires more time to change and react to a new condition. On the other hand, simplicity of the infrastructure does not remove the need to change or adjust. It just allows it to happen faster.

For example, you should keep this in mind when you are evaluating:

- The impact on the business from an unplanned outage or disaster
- Keeping up with planned maintenance
- Difficulty monitoring servers and business process flows
- Backing up

Reliability

The need to respond to the business needs pushes the IT environment to become more and more reliable. Not only is reliability an important part of the functionality of an IT architecture, it is crucial to the operating environment on

which the architecture is deployed. The following considerations are important when we look at reliability:

- Operating platform reliability and stability

  The IT must be available when the business needs them. Mission-critical application availability directly correlates to successful business operations. In today's on demand business environment, "downtime," whether planned or unplanned, is not only unwelcome—it's costly. Downtime statistics are staggering and range from tens of thousands of dollars to multiple millions of dollars per hour of downtime. And although the direct financial impact is significant, the damage can extend well beyond the financial realm into key areas of customer loyalty, market competitiveness, and regulatory compliance.

  Figure 1-1 on page 4 shows the financial impact of down time per hour on various kinds of industries.

Table 1-1 Financial impact of downtime per hour by industries

Brokerage Retail	$6.5 million
Credit Card Sales Authorization	$2.6 million
Airline Reservation Centers	$90,000
Package Shipping Services	$28,250
Manufacturing Industry	$26,761
Banking Industry	$17,093
Transportation Industry	$9,435
(Source: ©Eagle Rock Alliance, LTD. All Rights Reserved 2003)	

- Predictable and consistent performance

  Even when strong fluctuations in demand occur, the supporting IT systems must remain responsive. Infrastructures must be adequately provisioned to handle peaks that can be predicted, such as seasonal increases in transaction volume, as well as those that might catch the IT department by surprise.

- Integrity

  When integrating different systems that are based on different technologies and possibly controlled by partners outside the control of the company, integrity becomes very important. Maintaining integrity of information exchanged between partners and systems is needed to ensure that information has not been tampered with during the exchange. Also the integrity of the business transaction that spans all of these systems must be controlled, and this requires support for atomic transaction capabilities.

- Security

  The IT systems and infrastructure must provide the level of security in accordance with the requirements of the business processes.

Cost effectiveness

As the need to provide higher levels of service rise, budgets are tending to move in the opposite direction. IT executives are constantly challenged to provide better and additional service with smaller budgets.

Figure 1-2 shows the shift in IT costs between 1995 and 2004. In the early days of the IT industry the hardware was expensive and the need for processing power was often at a premium. Applications were written by specialists in low-level coding languages such as Assembler, COBOL, and others in order to use a minimal amount of processing power. The expenses for those IT developers were costly.

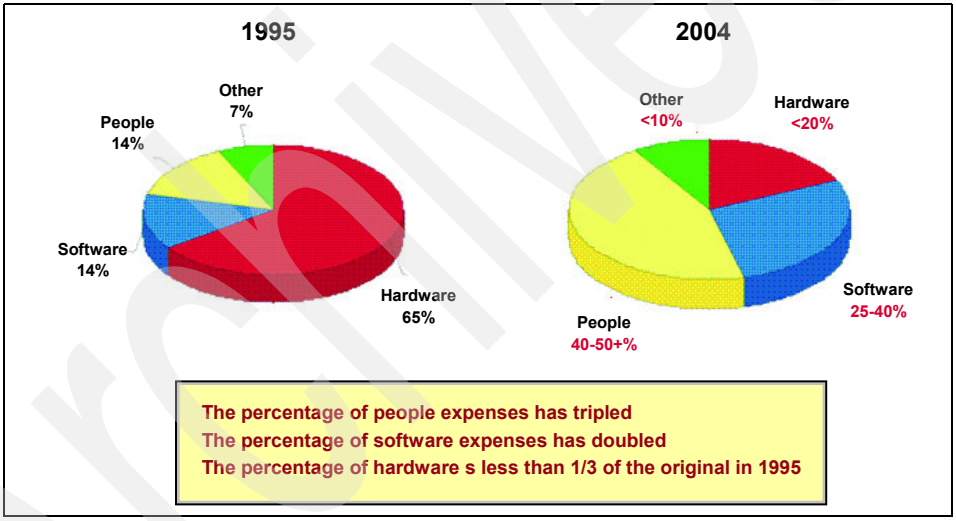


Figure 1-2 IT cost: 1995 and 2004

Today, IT can use high-level programming languages such as Java, Visual Designer, and others that might use more processing power but can be written by programmers who cost less to the businesses that hire them.

More hardware means using more energy, performing more hardware management, and in most cases deploying more servers with less than 50% resource utilization. Having this kind of unused or underused performance power and resources is not cost effective. Dealing with large numbers of servers with low utilization on each of one is a situation that most IT executives want to avoid.

New technologies based on sharing resources and data, plus the use of high-level programming languages, can help the IT architecture become more cost effective.

Cost of energy

One of the problems data centers face today and in the future is the cost of energy.

More servers require more and more processing power. Businesses continue to add more customers and applications to their IT environment, and that results in greater demands for processing power.

The demand to scale up and scale out has an effect on energy consumption. Many data centers that are trying to minimize energy costs for computing power are being stretched to the limit. These data centers often need to spend more resources on cooling and on power electrical demands.

1.3 The System z platform

As discussed in previous sections, infrastructure simplification is key to solving many IT problems. Simplification can be achieved by resource sharing among servers. It is all about sharing data, sharing applications, and simplified operational controls. The System z platform, along with its highly advanced operating systems, provides standard format, protocols, and programming interfaces that enable resource sharing among applications running on the mainframe or a set of clustered mainframes.

Resource sharing is intended to help reduce redundancy that often comes from maintaining multiple copies of duplicate data on multiple servers. Sharing can also improve privacy management by enabling better control and enforcing privacy regulations for data sources. Sharing data can help simplify disaster recovery scenarios because fewer servers are being deployed; therefore, sharing data means that less data needs to be protected during periodic back-up operations (for example, daily or weekly maintenance) compared to having multiple copies. But most of all, infrastructure simplification helps a business assess its entire computing capabilities to determine the best directions and strategy for overall, integrated workflow—and in doing so, helps to better leverage existing assets and drive higher returns on IT investments.

1.3.1 Using System Z technology to reduce complexity

System z servers offer capabilities that can help reduce the size and the complexity of a modern IT infrastructure. The ability to “scale up,” or add

processor power for additional workloads, is a traditional mainframe strength. Today's System z servers are available with up to 54 processors in a single footprint. Businesses can order a System z server configured with less than the maximum amount of processor power, and upgrade it on demand, which means using a customer-initiated procedure to add processing power when it is needed to support new applications or increased activity for existing applications, without waiting for a service representative to call.

Processing power can also be turned on (or activated) when needed and turned off when it is no longer needed. This is particularly useful in cases of seasonal peaks or disaster recovery situations.

Adding processing power and centralizing applications represents one strategy to help control the cost and complexity of an infrastructure. This approach can also provide a highly effective way to maximize control while minimizing server sprawl—in essence, reducing the number of single-application servers operating in uncontrolled environments. A number of single-application servers can typically be deployed to support business processes in both production and supporting test environments. Hot stand-by failover servers, quality assurance servers, backup servers, and training, development, and test servers are some of the types of resources that are required to support a given application. A System z server can help reduce the numbers of those servers by its ability to *scale out*.

The term "scale out" describes how the virtualization technology of the System z server lets users define and provision virtual servers that have all of the characteristics of distributed servers, except they do not require dedicated hardware. They coexist, in total isolation, sharing the resources of the System z server.

Virtual servers on System z can communicate between each other, using inter-server communication called HiperSockets™ (think of it as an "in-memory" TCP/IP network). This technology uses memory as its transport media without the need to go out of the server into a real network, thereby simplifying the need to use cables, routers, or switches to communicate between the virtual servers.

1.3.2 Business integration and resiliency

We have seen how the need to be flexible and responsive drives businesses. If the site is not up or responsive to its clients or employees when they need it, the more likely it will lose the customers, or it will take the employees more time to do their jobs. A resilient infrastructure and integrated applications are also critical to the success of any business.

Availability

One of the basic requirements for today's IT infrastructure is to provide continuous business operations in the event of planned or unplanned disruptions. The availability of the installation's mission-critical applications, based on a highly available platform, directly correlates to successful business operations.

System z hardware, operating systems, and middleware elements have been designed to work together closely, providing an application environment with a high level of availability. The System z environment approaches application availability with an integrated and cohesive strategy that encompasses single-server, multi-server, and multi-site environments.

The System z hardware itself is a highly available server. From its inception all of the hardware elements have always had an internal redundancy. Starting with the energy components and ending with the central processors, all of these redundant elements can be switched automatically in the event of an error. As a result of this redundancy, it is possible to make fixes or changes to any element that is down without stopping the machine from working and providing support for the customers.

The System z operating system that sits on top of the hardware has traditionally provided the best protection and recovery from failure. For example, z/OS, the flagship operating system of the System z platform, was built to mask a failure from the application. In severe cases, z/OS can recover through a graceful degradation rather than end in a complete failure. Operating system maintenance and release change can be done in most cases without stopping the environment.

Middleware running on z/OS is built to take advantage of both the hardware and operating system availability capabilities. IBM middleware such as IBM DB2® for z/OS, IBM CICS® products, IBM WebSphere® Application Server, and IBM IMS™ can provide an excellent solution for an available business application.

System z Parallel Sysplex architecture allows clustered System z servers to provide resource sharing, workload balancing, and data sharing capabilities for the IT, delivering ultimate flexibility when supporting different middleware applications. Although System z hardware, operating systems, and middleware have long supported multiple applications on a single server, Parallel Sysplex clustering enables multiple applications to communicate across servers, and even supports the concept of a large, single application spanning multiple servers, resulting in optimal availability characteristics for that application.

Parallel Sysplex is a cluster solution that is implemented from the IBM hardware to the middleware layer and, as a consequence, does not have to be designed and developed in the application layer.

With Parallel Sysplex and its ability to support data sharing across servers, IT architects can design and develop applications that have a single, integrated view of a shared data store. System z shared databases also provide high-quality services to protect data integrity.

This single-view database simplicity helps remove management complexity in the IT infrastructure. And simpler IT infrastructures help reduce the likelihood of errors while allowing planned outages to have a smaller impact across the overall application space.

Figure 1-3 shows the System z high availability family solution, from single system to the IBM Geographically Dispersed Parallel Sysplex™ (GDPS®).

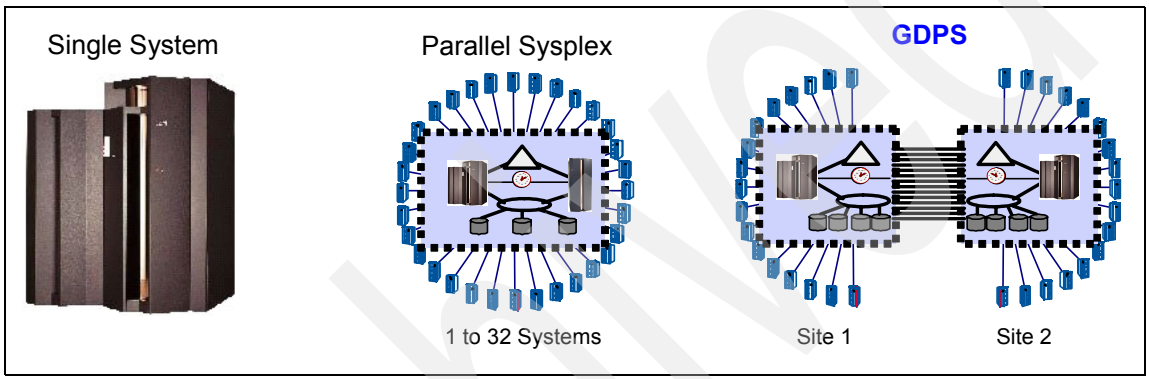


Figure 1-3 System z availability

GDPS technology provides a total business continuity solution for the z/OS environment. GDPS is a sysplex that spans multiple sites, with disaster recovery capability, based on advanced automation techniques. The GDPS solution allows the installation to manage remote copy configuration and storage subsystems, automate Parallel Sysplex operation tasks, and perform failure recovery from a single point of control.

GDPS extends the resource sharing, workload balancing, and continuous availability benefits of a Parallel Sysplex environment. It also significantly enhances the capability of an enterprise to recover from disasters and other failures, and to manage planned exception conditions, enabling businesses to achieve their own continuous availability and disaster recovery goals.

Hardware and software synergy

System z operating systems were designed to use central processors (CP). The vital connection between the hardware and the software resulted in the development of new instructions for the central processor that over time were able to respond to new application demands. The System z platform database

product DB2 for z/OS also makes use of the specialized instructions to speed up some basic database calculations.

The recently announced IBM System z Application Assist Processor (zAAP) is used by the z/OS Java Virtual Machine. z/OS can shift Java workloads to this new zAAP, thereby letting the CP focus on other non-Java workloads.

The IBM Integrated Facility for Linux (IFL) is another processor that enables the z/Linux operating system to run on System z hardware.

Recently, IBM announced yet another specialized processor, the IBM System z9 Integrated Information Processor (zIIP). The zIIP is designed to help improve resource optimization for running database workloads in z/OS.

New processors such as zAAP and zIIP lower the cost of the platform, making it more cost effective and reducing software costs.

1.3.3 Managing the System z platform to meet business goals

When new workloads are added to a System z server, they are not simply added randomly. Usually a workload is distinguished by its importance to the business. Some workloads, such as those associated with customer ordering and fulfillment, tend to have a higher degree of importance than applications used internally. Making resources available to mission-critical applications when they need them is a priority for System z hardware and software designers.

System z servers running a single z/OS image or z/OS images in Parallel Sysplex can take advantage of the Workload Manager (WLM) function. The overall mission of these advanced workload management technologies is to use established policy and business priorities to direct resources to key applications when needed. These policies are set by the user based on the needs of the individual business. These time-tested workload management features provide the System z environment with the capability to effectively operate at average utilization levels exceeding 70% and sustained peak utilization levels of 100% without degradation to high-priority workloads.

Figure 1-4 on page 12 shows the effect of CPU sharing on a System z server with multiple and different workloads running concurrently. In an environment not constrained for CPU, the response time for each application is not affected by the other applications running at the same time.

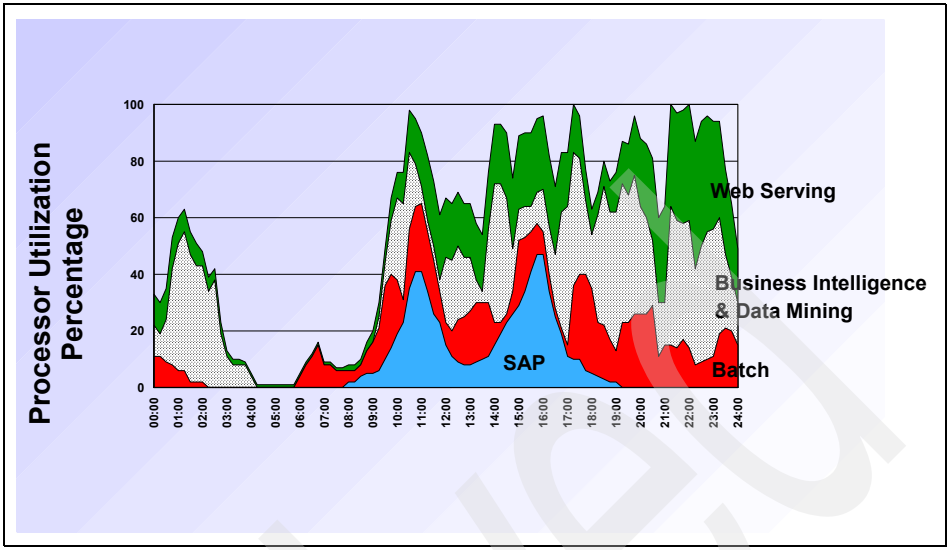


Figure 1-4 Mixed workloads on the System z platform

The higher degree of workload management represents a key System z advantage. Workload management can start at the virtual server level and drill down to the transaction level, enabling the business decide which transaction belonging to which customer has a higher priority over others.

The Intelligent Resource Director (IRD) is a technology that extends the WLM concept to virtual servers residing on a System Z server. IRD, a combination of System z hardware and z/OS technology that is tightly integrated with WLM, is designed to dynamically move server resources to the systems that are processing the highest priority work.

Important: All of these capabilities can be observed by running the Mettle Test. The Mettle test is a simulation of a realist z/OS production environment running WebSphere Application Server and traditional workloads with high transaction volume and high-availability configuration. The Mettle Test has been performed in a real environment in IBM Gaithersburg, and the test is a summary of the workload runs. For information about how to download and run the Mettle Test, refer to Appendix A, “Additional material” on page 215.

1.3.4 Security

For a business to remain flexible and responsive, it must be able to give access to its systems to existing customers and suppliers as well as to new customers, while still requiring the proper authorization to access e-commerce systems and

data. The business must provide access to the data that is required for the business transaction, but also be able to secure other data from unauthorized access. The business needs to prevent rogue data from being replicated throughout the system and to protect the data of the trusted partners. In summary, the business must be open and secure at the same time.

The System z environment, as with its previous mainframe models, has the security concept deeply designed in the operating system. The ability to run multiple applications concurrently on the same server demands isolating and protecting each application environment. The system has to be able to control access, allowing users to get to only the applications and data that they need, not to those that they are not authorized to use.

Hardware components, such as those for the cryptographic function implemented on each central processor, deliver support to the System z platform for encryption and decryption of data, and for scaling up the security throughput of the system.

In addition, other security components such as RACF (Resource Access Control Facility) provide centralized security functions such as user identification and authentication, access control to specific resources, and the auditing functions that can help provide protection and meet the business security objectives.

1.4 Summary

The System z platform with its rock-solid reliability, highly advanced scalability and integration capabilities, as well as its attractive cost efficiencies, can be a major player in today's IT infrastructure.

Business requirements such as flexibility, responsiveness to customer and market needs, and the ability to be cost effective are provided by the System z hardware and software.

In upcoming chapters we introduce a more technical view of the System z platform and point out how this platform plays a significant role in today's IT infrastructure.

2

System z architecture and hardware platform

In computer science the term *platform* usually defines the complex of hardware and software components that work together in order to provide applications with an execution environment.

System z is the name of the IBM platform designed to provide the most reliable environment for high-value critical enterprises applications and aimed at reaching a *near-zero* service downtime. The System z platform is commonly referred to as a mainframe.

In this chapter we give a technical overview of System z platform, explaining its architecture and its hardware.

2.1 History

The origin of mainframe computers dates back at least to the 1950s. In those days, mainframe computers were not just the largest computers; they were the *only* computers and few companies could afford them. Mainframe computers were single-purpose—some computers were for scientific processing, and other families of computers were used for business processing.

The introduction of the IBM System/360™ (or S/360™) in 1964 signaled the start of the new generation: the first general-purpose computers. In the decades since the 1960s, mainframe computers have steadily grown to achieve enormous processing capabilities.

In the summer of 1970, IBM announced a family of machines with an enhanced instruction set, called System/370™. The era of miniaturization had begun. In 1970 IBM rolled out a 128-bit bipolar chip that was used in the industry's first all-monolithic main memory, replacing the existing memories made from ferrite cores.

In September 1990, IBM announced System/390®. With System/390 IBM introduced a new mainframe technology implemented in Complementary Metal Oxide Silicon (CMOS) which used far less electricity, took up much less space, and cost less than the previous bipolar chip processors.

IBM unveiled in October 2000 the IBM eServer™ zSeries® 900 followed in May 2003 with the IBM eServer zSeries 990 as a very sophisticated server. The heart of the z990 is the IBM multi-chip module (MCM)—a very dense, advanced semiconductor and packaging technology. The new MCM was 50% smaller, enabling the z990 to deliver almost three times the processor capacity of the z900 in the same footprint. The module uses IBM's leading-edge copper and Silicon-on-Insulator (SOI) technology and contained over 3.2 million transistors.

In October 2003 IBM published the *Mainframe Charter*, which asserts its strong commitment to:

- Continue to provide leadership in innovation, maintaining System z position as a benchmark for flexible, efficient, and responsive platforms for highly complex, integrated environments running a wide range of mission-critical workloads.
- Enhance the value proposition and lower the cost of computing of System z solutions in a way that is compelling, clear, and consistent.
- Foster a community, supporting programs designed to foster vitality in the System z community and providing the skills and expertise to assist customers in designing, developing, and deploying on demand solutions built on a foundation whose cornerstone is the System z platform.

In April 2006 IBM announced the latest servers of the System z9 family:

- The System z9 Enterprise Class (z9 EC), which provides a strong combination of past mainframe characteristics, plus new functions designed around scalability, availability, and security and capable of fulfilling large enterprises' needs.
- The System z9 Business Class (z9 BC), which, while being technologically similar to z9 EC, is designed specifically as a midrange mainframe for delivering excellent price and performance for those customers requiring a lower-capacity entry point, in particular for small and medium businesses.

Early mainframe systems were housed in enormous, room-sized metal boxes or frames, which is probably how the term mainframe originated. Early mainframes required large amounts of electrical power and air-conditioning; larger models were water cooled. Most typical customer sites had several mainframes installed, with the I/O devices connected to all of the mainframes.

Modern mainframe servers are much smaller than earlier servers—about the size of a large U.S. refrigerator. The z9 EC is made up of two frames, and the z9 BC has just one frame. Each frame has a footprint of less than 3 sq ft (0.9 sq m) and is about 6.25 ft (1.9 m) high. Its power consumption ranges from 6.3 kW to 18.3 kW, depending on the configuration.

Figure 2-1 shows the IBM ES/9000® in 1992 and the System z9 EC in 2006.

Figure 2-1 IBM ES/9000 (1992) and System z9 EC (2006)

2.2 System z architecture

The System z architecture was designed on the concept of *sharing*. Sharing starts in the hardware components and ends with the data that is being used by

the platform. Before exploring System z architecture it is useful to discuss some basic key points.

2.2.1 Multiprogramming and multiprocessing

The earliest operating systems were used to sequentially control single-user computer systems. In contrast, current computer systems are capable of *multiprogramming* (executing many programs concurrently). When a job cannot use the processor, perhaps because it needs to wait for an asynchronous I/O operation to complete, multiprogramming enables the system to suspend the job, freeing the processor to work on another job. When the I/O operation completes, the currently executing piece of work is interrupted and the suspended job is made ready to run.

Operating systems make multiprogramming possible by capturing and saving status information about the interrupted program before allowing another program to execute. When the interrupted program is ready to begin executing again, it can resume execution just where it left off. Multiprogramming enables the operating system to run thousands of programs simultaneously.

Where a computer system has multiple processors, the operating system supports *multiprocessing*, where each of the multiple processors simultaneously processes separate instruction streams from the various programs. The multiple processors share the various hardware resources, such as memory and external disk storage devices. Multiprocessing enables multiple programs to be executed simultaneously, and allows a single program to use multiple processes in the same program to do parallel work.

The System z environment has supported multiprogramming and multiprocessing for its users for many years. Typical mainframe workloads include long-running database applications updating millions of database records, and online applications for thousands of interactive users.

Multiprocessing in z/Architecture

z/Architecture® defines that a single processor can process one, and only one, instruction stream at a time. The System z architecture is documented in the publication *z/Architecture Principles of Operation,* A22-7832-04. This book defines the System z hardware, including the instructions and the resources required to process them.

When you add more processors to the server, each processor shares the computer's memory and processes its stream of instructions simultaneously to, and independent of, the other processors. A number of z/Architecture instructions were designed to ensure integrity in a multi-processing environment. This is done so that z/OS can implement locking and other serialization

techniques to ensure that shared memory structures are modified with integrity. A single copy of the operating system manages all processing, and work is assigned to an available processor. If a processor fails, the z9 hardware will assign a spare processor and the work continues without a failure.

2.2.2 The virtualization concept

The ability to share everything is based on one of the major concepts of the System z mainframe: virtualization. As it is commonly used in computing systems, virtualization refers to the technique of hiding the physical characteristics of the computing resources from users of those resources. For example, each operating system in a logical partition of the System z platform thinks it has exclusive access to a real hardware image. The actual resources are shared with other logical partitions. The virtualization concept is not an add-on to the platform. z/Architecture contains many facilities in both the hardware and software that facilitate resource virtualization. All elements of System z enable the virtualization concept.

Virtualizing the System z environment involves creating virtual systems (logical partitions and virtual machines), and assigning virtual resources, such as memory and I/O channels, to them. Resources can be dynamically added or removed from these logical partitions though operator commands. The customer can also use a facility called Capacity on Demand, which allows new hardware resources to be added to the z9 non-disruptively.

Dynamic logical partitioning and virtual machines increase flexibility, enabling selected system resources such as processors, memory, and I/O components to be added and deleted from partitions while they are actively in use. The ability to reconfigure logical partitions dynamically enables system administrators to dynamically move resources from one partition to another.

Processor Resource/Systems Manager™ (PR/SM™) is a hypervisor integrated with all System z elements that maps physical resources into virtual resources so that many logical partitions can share the physical resources.

2.2.3 PR/SM and logical partitions

Processor Resource/Systems Manager is a feature of System z9 and previous IBM mainframes that enables logical partitioning of the central processor complex (CEC). This feature can be used to aggregate physical system resources into shared pools from which virtual systems receive virtual resources. This enables the user to consolidate workloads and operating environments currently running on separate processors into one physical system while using resource sharing to improve resource utilization and maintain performance.

PR/SM provides the logical partitioning function of the central processor complex (CPC). It provides isolation between partitions, which enables installations to separate users into distinct processing images, or to restrict access to certain workloads where different security clearances are required.

Each logical partition operates as an independent server running its own operating environment. On the latest System z models, you can define up to 60 logical partitions running z/VM®, z/OS, Linux on IBM System z, z/TPF, and more operating systems. PR/SM enables each logical partition to have dedicated or shared processors and I/O channels, and dedicated memory (which you can dynamically reconfigure as needed).

In other words, PR/SM transforms physical resources into virtual resources so that several logical partitions can share the same physical resources. Figure 2-2 shows the PR/SM and LPAR concept.

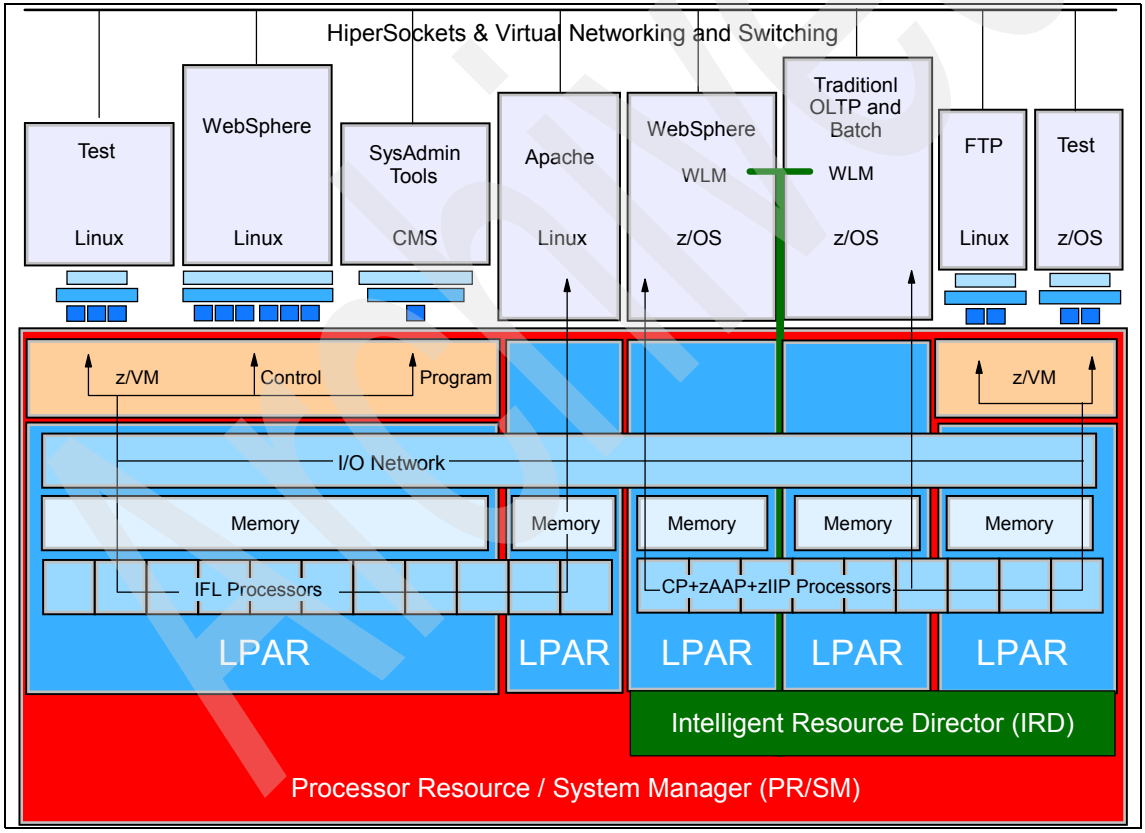


Figure 2-2 PR/SM and LPAR concept

2.3 System z hardware architecture

System z, as with all computing systems, is built on hardware components. Most of those components were introduced early in the mainframe era, and were developed over the years. Figure 2-3 is a simplified schematic diagram of the System z hardware components that are described in the following paragraphs.

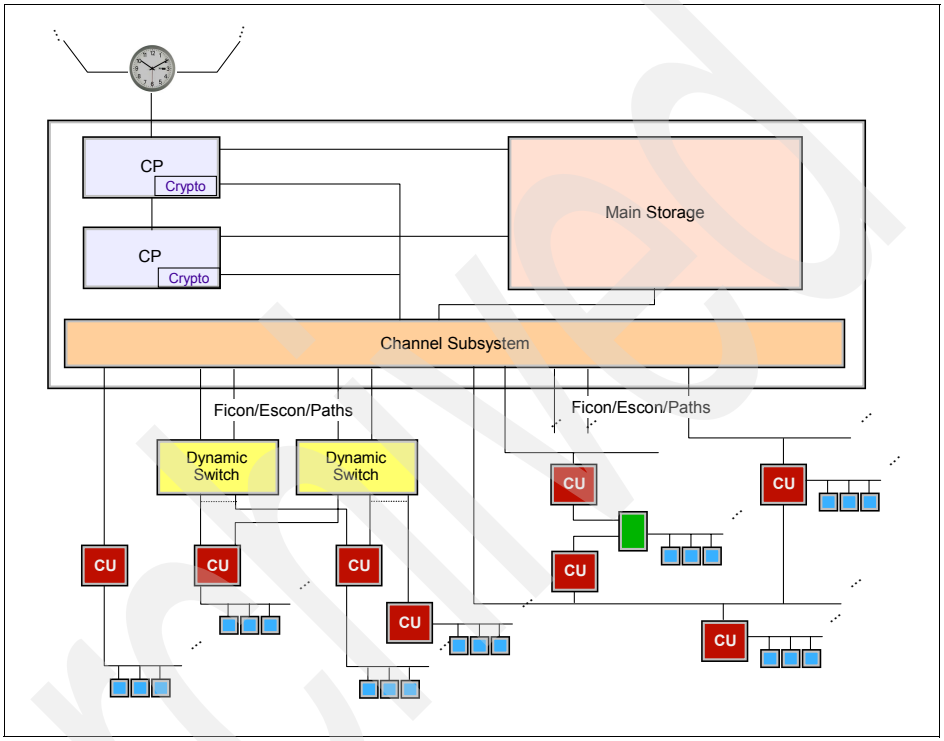


Figure 2-3 System z basic components

2.3.1 The processing unit

The processing unit (PU) is the heart of every computer hardware. The System z platform has a unique processing unit with its own set of instructions. There are different types of processing units, but all of them are built using the same chip and characterized as a specific processor type. If a processor is uncharacterized. it is available as a spare central processor to automatically take over the processing if a hardware failure occurs in any other running processing unit.

In the System z EC model there could be up to 54 processors. Current System z processing unit types include:

- Central processor (CP)
  This is the basic processor, available for all kinds of operating systems and application software.
- Integrated Facility for Linux (IFL)
  This is a CP that can run only the Linux operating system or z/VM in support of Linux. The System z model number is used for much of the software license charge billing. An IFL processor is not counted when specifying the model number of the machine, so the processing capacity supplied by the IFL does not affect software costs for the general purpose CPs. This can make a substantial difference in software costs to the customer.
- System z Application Assist Processor (zAAP)
  A zAAP processor is used by the IBM Java Virtual Machine (JVM™) to execute Java code. As with IFL engines, zAAP engines do not contribute to the software prices charged for the general purpose CPs.
- System z9 Integrated Information Processor (zIIP)
  The System z9 Integrated Information Processor (zIIP) is an engine for processing eligible database workloads. The zIIP is designed to help lower software costs for select workloads on the mainframe, such as business intelligence (BI), enterprise resource planning (ERP), and customer relationship management (CRM). Currently the only product from IBM that supports re-directing some of its work to a zIIP is DB2 V8; however, the API to use the zIIP is available to other Independent Software Vendors (ISVs). Also, as with previous processors, the zIIP does not count when specifying the model number of the system, and so adds capacity to the System z platform without adding additional software charges.
- Integrated Coupling Facility (ICF)
  These processors can only run Coupling Facility Control Code (CFCC). They are not visible to normal operating systems or applications. This processor is used to create a Coupling Facility for use by the System z Parallel Sysplex solution.
- Spare
  An uncharacterized processing unit functions as a “spare.” If the system controllers detect a failing processing unit, it can be replaced with a spare. In most cases this can be done without any system interruption, even for the application running on the failing processor.
- Various forms of *capacity on demand* and similar arrangements exist whereby a customer can enable additional processor (CP, IFL, zAAP, or zIIP) at certain times (for unexpected peak loads, for example).

Note: The concept of capacity on demand and the CPs used for it are explained in Chapter 5, “Resiliency” on page 97.

On each System z machine there is another processor that is used by the System z hardware and not directly by any System z operating systems. The system assist processor (SAP) uses the same processing unit hardware as the previous processors, but executes internal code and is a part of the I/O subsystem of the server. The SAP acts as an Integrated Offload Processor (IOP) engine, executing the channel programs that perform the I/O operations for the platform. In a System z configuration there can be one or more SAPs.

2.3.2 Memory

Earlier in this chapter we discussed the concepts of multiprogramming and multiprocessing. Another concept that is based on multiprogramming is the *multithreading* or, as it is called in the mainframe world, *multitasking*. Multitasking is done explicitly by the computer programmer. The program is structured and written to be divided into different tasks that can run in parallel on different processors.

Many users running many separate programs means that, along with large amounts of complex hardware, z/OS needs large amounts of memory to ensure suitable system performance. Large companies run sophisticated business applications that access large databases and industry-strength middleware products. Such applications require the operating system to protect privacy among users, as well as enable the sharing of databases and software services.

Thus, multitasking and the need for a large amount of memory mean that z/OS must provide functions beyond simple, single-user applications. The sections that follow explain, in a general way, the attributes that enable z/OS to manage complex computer configurations.

In System z terminology, the concept of memory is called *storage*. The System z platform uses two types of memory (storage):

- *Real storage* (also called *central storage*), located on the System z hardware itself. Instructions accessing real storage will appear to have accessed the memory synchronously with the processor. That is, certain instructions must be delayed while data is retrieved from central memory. Both data and programs must be loaded into central storage (from input devices) before they can be processed by the CP.
- *Auxiliary storage*, (also called *external storage*) located external to the System z hardware. Auxiliary storage is accessed asynchronously, through an input/output (I/O) request, where one or more pages of memory are written

or read. During an I/O request, the processor is free to execute other, unrelated work.

Inside the System z hardware we can find the physical memory. This is not the real memory, but has a major role in creating it. Current System z models can have up to 512 GB of physical memory. The amount of physical memory can be increased concurrently. Real memory is built over the physical memory. The System z9 provides physical memory in four identically sized storage cards. The real memory assigned to a logical partition is assigned from all four cards, so that the memory access can be “interleaved” or striped over the four cards. This decreases the average time to fetch memory from the real storage, because four fetch operations occur simultaneously. Figure 2-4 describes this concept.

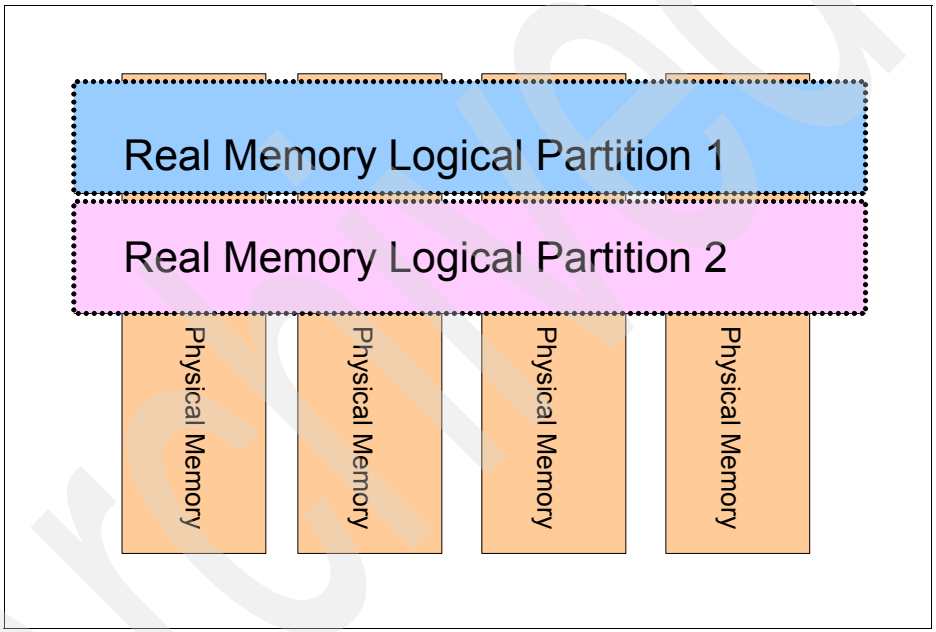


Figure 2-4 Real and physical memory assignment

Note: In the past, when memory was expensive, another type of memory was built on top of physical memory, called *expanded storage*. This memory was built by using physical memory that was slower and cheaper than the physical memory used for the central memory. Expanded storage was addressable only at page level, so instructions could not be executed from expanded storage. z/OS no longer uses expanded storage. The z/VM operating system uses expanded memory as a high-speed paging device. Expanded storage is now implemented using the same physical memory as the central memory.

Virtual memory

The System z environment uses both types of memory (real and auxiliary) to enable another kind of memory called *virtual storage*. In z/OS, each user has access to virtual memory storage, rather than real memory.

The virtual memory is a combination of a portion of the real memory and a portion of the external memory. The data sets on the external memory that are part of the virtual storage are called page data sets. Figure 2-5 shows the concept of System z virtual memory.

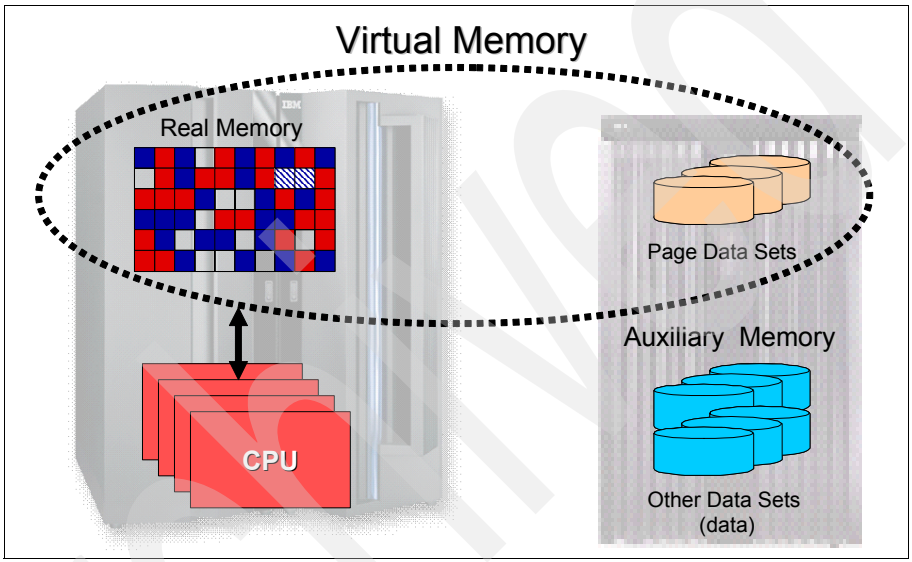


Figure 2-5 Virtual memory in System z

Currently, the architecture supports 64-bit long addresses, which enables a program to address up to 16 exabytes. In reality, the mainframe will have much less central memory installed. How much less depends on the model of the computer and the system configuration. To enable each user to act as though this much memory really exists in the computer system, z/OS keeps only the active block of data of each program in real memory. *Working set* is the technical term to indicate the pages of virtual memory that stay active during a time interval and are kept in real memory/central storage.

The pieces of a program executing in virtual memory must be moved between real and auxiliary memory. To allow this, z/OS manages memory in units, or page, of 4 KB. The following terms are defined:

- A block of central storage is a *frame*.
- A block of virtual memory is a *page*.
- A block of auxiliary storage is a *slot*.

An active virtual storage page resides in a central storage frame. A virtual memory page that becomes inactive is written out to an auxiliary memory slot (in a paging data set).

Memory addressing

For purposes of addressing the System z memory, three basic types of addresses are used: *absolute*, *real*, and *virtual*. The addresses are distinguished on the basis of the transformations that are applied to the address during a memory access. Address translation converts virtual to real, and prefixing converts real to absolute.

Dynamic address translation is the process of translating a virtual address during a memory reference into the corresponding real address.

As a consequence, the System z operating systems use a series of tables and indexes to relate locations on auxiliary memory to locations in central memory, and special settings (bit settings) to keep track of the identity and authority of each user or program. These tables are defined in the *z/Architecture Principles of Operation,* A22-7832-04.

2.3.3 The channel subsystem

One of the main strengths of the mainframe computers is the ability to deal with a large number of simultaneous I/O operations. The channel subsystem (CSS) has a major role in providing this strength. The CSS manages the flow of information between I/O devices and central memory. By doing so it relieves CPUs of the task of communicating directly with I/O devices and permits data processing to proceed concurrently with I/O processing.

The channel subsystem is built on the following components:

SAP (system assist processor) One of the System z processor types, the SAP connects the channel subsystem to the I/O devices that are attached to the channels. The SAP uses the I/O configuration loaded in the Hardware System Area (HSA), and knows which device is connected to each channel, and what is its protocol. The SAP manages the queue of I/O operations passed to the channel subsystem by z/OS.

Channels These are small processors that communicate with I/O control units (CU). They manage the transfer of data between central storage and the external device.

Channel path The channel subsystem communicates with I/O devices through channel paths. If a channel is shared between multiple logical

partitions, each logical partition establishes a unique channel path to the each device using this channel.

Subchannels A subchannel provides the logical appearance of a device to the program and contains the information required for performing a single I/O operation. One subchannel is provided for each I/O device addressable by the channel subsystem.

Physical channel types

The channel subsystem may contain more than one type of channel path:

ESCON® Enterprise Systems Connection. Since 1990, ESCON has replaced the S/370™ parallel channel as the main channel protocol for I/O connectivity, using fiber optic cables and a new “switched” technology for data traffic.

FICON® Based on the Fibre Channel Protocol (FCP), it was introduced in 1998. Because it is much faster than ESCON, it is becoming the most popular connection type.

OSA-2 Open Systems Adapter-2. A networking type of connector, it can be configured to support various network protocols, such as Ethernet, Fast Ethernet, token-ring, and Fiber Distributed Data Interface (FDDI).

OSA Express A faster networking connector that supports fast Ethernet, Gigabit Ethernet, and 10 Gigabit Ethernet.

ISC Inter System Communication (ISC) channels provide connectivity between the CF and the systems that are directly attached to it. It, and a CF are required for data sharing in a Parallel Sysplex.

ICB Integrated Cluster Bus. Very high speed copper links that connect to the CF.

The total number of channel paths provided by a logical channel subsystem depends on the model and on the configuration; the maximum addressability is 256 per Channel Set.

Logical Channel Subsystems

The Logical Channel Subsystem (LCSS) concept was introduced with the z990 server. The LCSS was created to enable the System z environment to handle more channel paths and devices available to the server, and to increase the number of logical partitions. Each LCSS may have from 1 to 256 channels, and may in turn be configured with 1 to 15 logical partitions. You can have two LCSSs on the z9 BC machine and four LCSSs on the z9 EC machine. When configuring four LCSSs the maximum device addressability is 1024.

2.3.4 Control units and I/O devices

The last two elements on the I/O chain are the control units (CU) and the I/O devices. The control unit provides the logical capabilities necessary to operate and control an I/O device and adapts the characteristics of each device so that it can respond to the standard form of control provided by the channel subsystem. An I/O device can be a disk, tape, network adapter, or other external device.

2.3.5 Time-of-Day clock

There is a longstanding requirement for accurate time and date information in data processing. As single operating systems have been replaced by multiple, coupled operating systems on multiple servers, this need has evolved into a requirement for both accurate and consistent clocks among these systems.

Server Time Protocol is a server-wide facility, implemented in the System z microcode, designed to provide the capability of time synchronization between the latest System z machines (z9 EC, z9 BC, z990, and z890).

STP is designed for servers that are configured in a Parallel Sysplex or a standard sysplex without a Coupling Facility, as well as servers that are not in a sysplex but need to be time-synchronized. STP supports a multi-site timing network of up to 100 Km (62 miles) over fiber optic cabling. STP is a message-based protocol in which timekeeping information is passed over data links between servers. The timekeeping information is transmitted over externally defined coupling links (ISC or ICB).

Note: STP is a chargeable feature that is the follow-on to the Sysplex Timer® device (IBM 9037).

2.3.6 CP Assist for Cryptographic Function

The CP Assist for Cryptographic Function (CPACF) is incorporated into every central processor of the System z server family. The CPACF feature delivers cryptographic support on every Central Processor (CP) with Data Encryption Standard (DES), Triple DES (TDES), AES data encryption/decryption along with MAC generation, secure hashing, and Pseudo Random Number Generation algorithms.

2.3.7 HiperSockets

HiperSockets is a technology that provides high-speed TCP/IP connectivity between servers running in different logical partitions (LPARs). The

communication is through the system memory of the processor. The virtual servers that are connected and form a “virtual LAN.”

2.3.8 Hardware Management Console (HMC)

System z hardware can be managed by using either the Support Elements directly attached to the server or using the Hardware Management Console (HMC), which is a desktop application providing the end user interface to control and monitor the status of the system.

Working from an HMC, an operator, a system programmer, or IBM technical personnel can perform basic operations on the System z servers. The HMC is a powerful console that can load and shut down system images as well as the entire machine. Some of the common capabilities of the HMC are:

- Load the System z hardware configuration.
- Load or reset a system image.
- Add and change the hardware configuration (most of them dynamically).
- Access the hardware logs.

All these functions can be executed using a browser interface in a totally secure environment.

2.3.9 Intelligent Resource Director (IRD)

Intelligent Resource Director, a feature of System z hardware and the Parallel Sysplex, is designed to further extend the lead of the System z environment in managing multiple heterogeneous workloads with various business priorities toward achieving their goals. Through its use, a more synergistic relationship is established between z/OS and System z hardware with respect to the allocation of resources among logical partitions.

For a more detailed discussion about IRD, see the IBM Redbook *z/OS Intelligent Resource Director,* SG24-5952.

WLM is responsible for enabling business-goal policies to be met for the set of applications and workloads. IRD implements the adjustments that WLM recommends to local sysplex images by dynamically taking the hardware (CPU and channels) to the LPAR where it is most needed. IRD has three main areas of responsibility:

- LPAR CPU management
- Channel Subsystem priority queueing
- Dynamic channel path management

LPAR CPU management

One of the resources that can be allocated to a logical partition is the processor weight of the partition. The LPAR weight determines the portion of the machine's shared CPU resources that are allocated to a logical partition when there is competition for CPU among two or more partitions. Without IRD it would be the job of the system programmer to set processor weight so that the workloads running in the logical partition receive the CPU they require. If the workload mix changes, the weight must be re-evaluated and manually changed.

The goal of the LPAR CPU management is to help simplify the configuration task by automatically managing physical processor resources to allow a high utilization of physical CPU capacity, while allowing performance objectives to be met at times of peak demands. IRD LPAR CPU management extends WLM goal-oriented resource management to allow for dynamic adjustment of logical partition processor weight. This function moves CPU to the partition with the most deserving workload, based on the WLM policy, and enables the system to adapt to changes in the workload mix. WLM keeps the total processor weight of the sysplex's LPAR constant, so that partitions that are not part of the sysplex are not affected.

Channel Subsystem Priority Queueing

Prioritizing I/O requests is not new to z/OS. In the early versions of the mainframe operating system, I/O requests could be prioritized on device queues within the operating system.

z/OS in WLM uses this new function to dynamically manage the channel subsystem priority of I/O operations for given workloads based on the performance goals for these workloads as specified in the WLM policy. In addition, because Channel Subsystem I/O Priority Queueing works at the channel subsystem level, and therefore affects every I/O request (for every device, from every LPAR) on the machine, you can also specify a single channel subsystem I/O priority that is to be used for all I/O requests from systems that do not actively exploit Channel Subsystem I/O Priority Queueing.

Dynamic channel path management (DCM)

Dynamic channel path management is designed to dynamically adjust the channel configuration in response to shifting workload patterns. It is a function in IRD, together with WLM LPAR CPU Management and Channel Subsystem I/O Priority Queueing.

DCM can improve performance by dynamically moving the available channel bandwidth to where it is most needed. Prior to DCM, you had to manually balance your available channels across your I/O devices, trying to provide sufficient paths to handle the average load on every controller. This means that at

any one time, some controllers probably have more I/O paths available than they need, while other controllers possibly have too few.

Another advantage of Dynamic channel path management is that you don't have to be as concerned about how busy you should run each of your channels or control unit types you have installed. Instead, you just need to monitor for signs of channel overutilization (high channel utilization combined with high Pend times).

2.4 System z frames and cages

The current System z hardware layout is the result of the continuous evolution of the mainframe hardware architecture. Over the years new components have been added to the design but the basic requirement had not been changed. The current design is highly available and has an inside redundancy for most of the parts.

System z hardware can be built on two frames (frame A and frame Z). The z9 BC has one frame; z9 EC has two frames. Not all of the frames are fully populated. Each frame can contain two cages and can also hold up to two Internal Battery Features (IBF) and up to two Modular Refrigeration Units (MRU).

There are two types of cages:

1. CEC cage: The basic cage, which contains the processor units (PU), the physical memory, and connectors to the other cages. On each System z machine there is only one CEC cage.
2. I/O cage: Contains the channels card that connects the System z machine to its external I/O devices. There are different channel path types (see "Physical channel types" on page 27). Each System z configuration has at least one I/O cage with a maximum of three I/O cages.

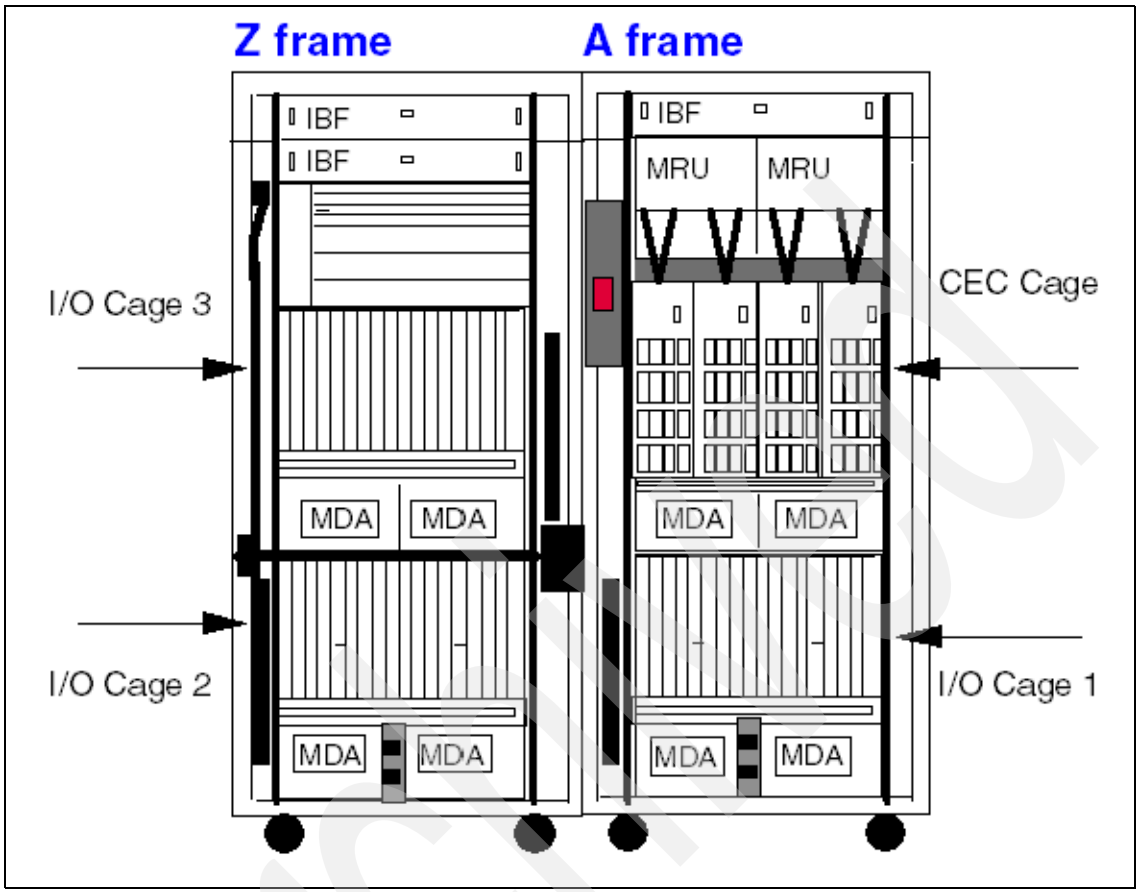


Figure 2-6 Frames and cages in the System z hardware

Optional battery feature (IBF)

IBF keeps the server powered up for up to 13 minutes when there is a power outage. This time amount is dependent on the number of I/O cages.

Cooling

System z hardware has a redundant air-cooled system assisted by refrigeration that is based on closed-loop cooling subsystem. The air system is based on the MRU (Modular Refrigerator Unit) and MDA (Motor Drive Assembly).

2.4.1 System z book

The book concept was first introduced on the z990 machine. A book contains processors, memory, and connection to the I/O cages. Books are located in the CEC cage. Each System z configuration has one to four books.

Each book contains:

- 12 to 16 processor units (PU)
- 16 GB to 128 GB of physical memory
- Up to eight memory bus adapters that are responsible for connectivity to the I/O cages

Important: The current maximum hardware configuration allows 54 processor units to be available to the operating systems. Physically there are up to 64 PUs installed on the machine. The 10 spare PUs are SAPs and spare processors for redundancy.

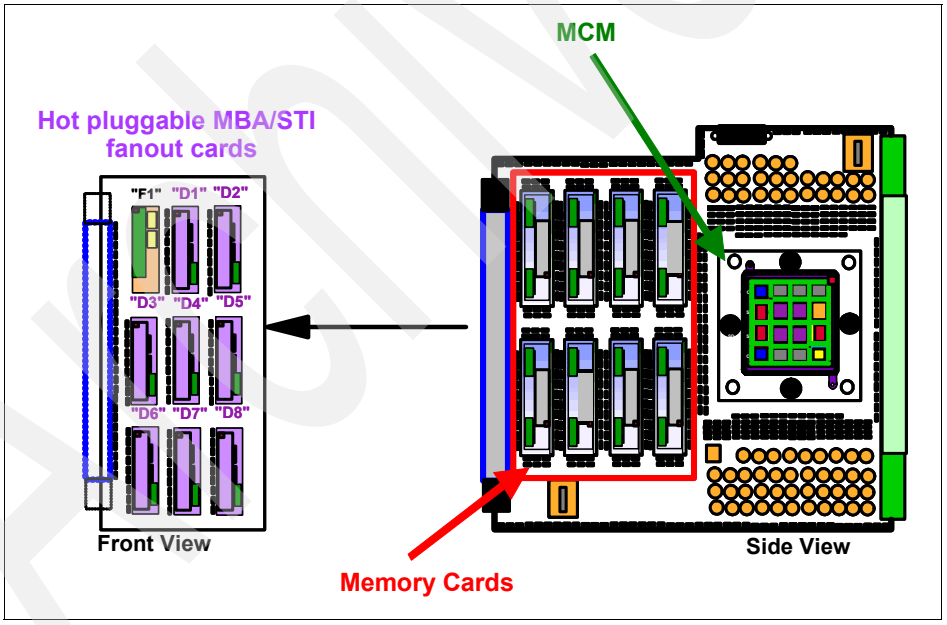


Figure 2-7 Book structure and components

Power

The power elements in System z hardware are designed with redundancy. It is recommended that each power element get its power supply from two different

power sources. Each book gets its power from two different power elements. Those power elements provide the required power for the book in an N +1 design, which means that there is more power capacity than is required by the book. If one of the power elements fails, the power requirement for the book is still satisfied from the remaining power elements.

2.5 The System z hardware models

The z9 EC has five models. The model numbers are directly related to the maximum number of PUs that can be characterized by the installation. For customer use, PUs can be characterized as CPs, IFLs, ICFs, zAAPs, zIIPs, or if needed, additional system assist processors (SAPs).

- The z9 EC Model S08 has one book with 12 PUs, of which eight can be characterized. The four remaining PUs are two SAPs and two spares.
- The z9 EC Model S18 has two books with 12 PUs in each book for a total of 24 PUs, of which 18 can be characterized. The six remaining PUs are four SAPs (two in each book) and two spares.
- The z9 EC Model S28 has three books with 12 PUs in each book for a total of 36 PUs, of which 28 can be characterized. The eight remaining PUs are six SAPs (two in each book) and two spares.
- The z9 EC Model S38 has four books with 12 PUs in each book for a total of 48 PUs, of which 38 can be characterized. The 10 remaining PUs are eight SAPs (two in each book) and two spares.
- The z9 EC Model S54 has four books with 16 PUs in each book for a total of 64 PUs, of which 54 can be characterized. The 10 remaining PUs are eight SAPs (two in each book) and two spares.

Whether one, two, three, or four books are present, to the user all books together appear as one Symmetric Multi Processor (SMP) with a certain number of CPs, IFL, ICFs, zAAPs, and zIIPs, and a certain amount of memory and bandwidth to drive the I/O channels and devices.

3

System z software

In this chapter we give a technical overview of operating systems that are available for the System z platform, with a focus on z/OS, the flagship operating system for the platform.

For z/OS, we describe some highlights of the software design, some of the major components, and available middleware.

3.1 Operating systems

This section describes the operating systems that are available for the System z platform with their characteristics and strengths. More details about z/OS will follow in later sections.

3.1.1 z/OS

z/OS, the flagship operating system of the System z platform. is able to exploit fully the capabilities of the System z hardware. As it exists today, z/OS is the result of decades of technological advancement. It evolved from an operating system that could process a single program at a time to an operating system that can handle many thousands of programs and interactive users concurrently.

z/OS is designed to deliver the highest quality of service for enterprise transactions and data, and extends these qualities to new applications using the latest software technologies. It provides a highly secure, scalable, high-performance base on which to deploy SOA-enabled applications using Internet and Java technologies. It also runs the traditional applications, which are part of the bricks customers want to reuse in their SOA environment.

In most early operating systems, requests for work entered the system one at a time. The operating system processed each request or *job* as a unit, and waited for its completion to start the next job. This arrangement worked well when a job could execute continuously from start to completion. But often a job had to wait for information to be read in from, or written out to, a device such as a tape drive or printer. Input and output (I/O) take a long time compared to the electronic speed of the processor. When a job waited for I/O, the processor was idle. Keeping the processor working while a job waited could increase the total amount of work the processor could do without requiring additional hardware. Multiprogramming enables multiple jobs to run concurrently, overlapping the idle time in one job with busy time in another. If there are multiple CPs, the jobs can run simultaneously.

Some workloads can use multitasking: dividing the single job into multiple tasks, each competing for processor time. With new workloads, the work may be processed in many *enclaves*, which are short pieces of work. z/OS will dispatch all of the tasks and enclaves, giving these work items to various system components and subsystems that function interdependently. At any point in time, one component or another gets control of the processor, makes its contribution, and then passes control along to a user program or another component.

Besides z/OS, there is a low-cost version of z/OS named z/OS.e that can be used for running only non-traditional workloads on the smaller System z hardware platforms.

3.1.2 z/VM

z/VM is an IBM z/Architecture (64-bit) operating system for the System z platform. The z/VM implementation of IBM virtualization technology provides the capability to run other full-function operating systems, such as Linux and z/OS, under z/VM as “guest” systems. z/VM supports 64-bit z/Architecture guests and 31-bit IBM Enterprise Systems Architecture/390 guests.

z/VM provides each user with an individual working environment known as a *virtual machine*, which appears to a guest operating system as hardware, but is simulated by z/VM and the System z hardware. The virtual machines under z/VM share the total system resources. Processor and memory capacity is designed to be allocated to the servers that need it, when they need it. The virtual machine simulates the existence of a dedicated real machine, including processor functions, storage, and I/O resources.

z/VM can create a virtual machine environment that can be functionally richer than a “real” environment. Examples of this include:

- Using data-in-memory techniques to transparently enhance guest system performance. Using these techniques a guest machine uses a virtual disk device that is simulated in real storage.
- Simulating processor, device, memory, and networking facilities that do not exist in the real mainframe system.
- Sharing a single copy of program code, including an operating system kernel among several guest systems.
- Running multiple virtual servers concurrently on a single processor, which is sometimes referred to as *sub-CPU granularity*.

z/VM lets you share disk space among virtual Linux images. Code can be shared read-only among the Linux guests on a System z server. This is a more efficient use of disk investment that can also provide version control for your application software and help to reduce the complexity of maintaining that software.

Operating-system-to-operating-system communication brings out another strength of a VM-based solution compared to using standalone systems. In a standalone environment, real cables must be connected to each server. By providing the guest access to a virtual network interface, z/VM provides “virtual” cables for your Linux servers. These virtual cables are very fast, providing memory-to-memory data transfer rates.

3.1.3 Linux on System z

Linux on System z supports the System z processor architecture and devices that are specific to System z environments. There are two supported distributions of Linux that can run on System z hardware:

- Novell SUSE
- Red Hat

Linux on System z can benefit from System z specific hardware:

- Cryptographic support: PCICA, CPA, PCIXCC, Crypto2.
- Traditional and open I/O subsystems.
- Disk (mainframe or SCSI) and tape.
- OSA-Express and OSA-Express2 for very high speed communication between z/OS, z/VSE™, z/TPF, and Linux.
- HiperSockets for ultra-high speed communication between z/OS, z/VSE, and Linux. HiperSockets is a hardware feature of the System z environment that provides high-speed TCP/IP connection between logical partitions.

Linux on System z automatically inherits the strengths and reliability features of the mainframe hardware. Indeed, Linux on System z, used in conjunction with z/VM, provides an efficient and cost-effective basis for server consolidation.

3.1.4 Other supported operating systems

In addition to the operating systems described above, the z/Transaction Processing Facility (z/TPF) and the z/Virtual Storage Extended (z/VSE) operating systems can run on mainframe hardware. The z/TPF operating system is a special-purpose system that is used by companies with very high transaction volumes, such as credit card companies and airline reservation systems. z/VSE is an operating system that can be used for smaller mainframe servers.

3.2 z/OS software design

In this book we focus on z/OS, a widely used mainframe operating system. z/OS is designed to offer a stable, secure, and continuously available environment for applications running on the mainframe.

z/OS is the result of decades of technological advancement. The current release is Version 1.8, which was made available in September 2006.

In this section we discuss some of the major z/OS design elements.

3.2.1 Address space

The range of virtual addresses that the operating system assigns to a working unit is called an *address space*. A working unit can be a user, a program, a batch job, and more. The address space, a range of virtual addresses, is available for executing instructions and storing data. z/OS provides each working unit with a unique address space and maintains the distinction between the programs and data belonging to each address space. Within each address space, the working unit can spawn multiple tasks, represented by *task control blocks* or *TCBs* that allow multitasking.

Virtual addressing permits the total addressable memory to be greater than the central memory capabilities of the system. The use of multiple virtual address spaces in z/OS provides virtual addressing capability to each job in the system by assigning each job its own separate virtual address space. The potentially large number of address spaces provides the system with a large virtual addressing capacity. Some parts of the address spaces are common to all address spaces.

With multiple virtual address spaces, errors are usually confined to one address space—a program running cannot see the storage in another address space's memory. This improves system reliability and makes error recovery easier. Programs in separate address spaces are protected from each other. Isolating data in its own address space also protects the data.

z/OS uses many address spaces. There is at least one address space for each working unit in progress and one address space for each user logged on through TSO, Telnet, rlogin, or FTP. Users working on z/OS through a major subsystem, such as CICS or IMS, use address spaces belonging to the subsystem, not their own address spaces. There are also many address spaces for operating system functions, such as operator communication, automation, networking, and security.

Private and common areas

The virtual address space is divided into areas according to the purpose of their use. Virtual memory allocation in each address space is divided between the system's requirements and the user's requirements. The system control code, and some system data, is mapped to identical virtual addresses in every address space. The pages for this common data are shared by all address spaces.

The private area consists of memory areas that are used by the programs that run in this address space and consist of program data and executable code. The common areas are memory areas that are common to all address spaces running on that z/OS system image. These areas are used by the operating system or to transfer data between address spaces.

To protect the different parts of the address space z/OS assigns each page a storage protection key. Storage cannot be modified unless the key of the current process matches the key of the storage or the process is running in key 0. The storage protection key can also indicate that the data cannot be fetched without a matching key. This convention ensures that a general user cannot accidentally overlay system or subsystem code.

Storage protection

The use of virtual storage addressing enables z/OS to maintain the distinction between the programs and data belonging to each address space. The private areas in one user's address space are isolated from the private areas in other address spaces, and this provides much of the operating system's security. This is because each user's private area is mapped to different real storage frames.

Data in central storage is then protected from unauthorized users by a doubled protection mechanism: the protection key and the fetch bit.

A control field called a protection key is associated with each 4 KB frame of central storage. When a request is made to modify or read the contents of a central memory location, the key associated with the request is compared to the memory protect key and access is allowed if both keys match or the accessing key is the supervisor key. At the same time the status of the fetch bit in that frame is checked. According to the status of both the protection key and the fetch bit, checking the request is approved or an exception interrupt occurs.

User address spaces are normally using storage key 8 while some system address spaces can use keys 1-7 for additioanl protection to ensure system services will not be corrupted by user programs.

Data spaces

A data space is a type of address space with a range up to 2 GB of contiguous virtual memory. The virtual memory map of a data space is quite different from the regular address space memory map. The whole 2 GB can be used for user data. Some models cannot use the first 4 KB of the data space. A data space can hold only data; program code cannot execute in a data space. A program can reside in a data space as data.

A program references data in a data space directly, in much the same way it references data in an address space. It addresses the data by the byte, manipulating, comparing, and performing arithmetic operations.

3.3 z/OS components

The z/OS design is based on some system components that help the operating system perform its tasks. In this section we discuss some of the system managers and system components.

3.3.1 Memory managers

To manage different kinds of memory, the z/OS uses three different memory managers:

- *Virtual Storage Manager*™ (VSM): Manages the virtual memory. Its main function is to allocate and free virtual storage. It works with RSM™ in allocating the virtual storage management tables.
- *Real Storage Manager* (RSM): Controls the allocation of the real memory page frames to back the virtual storage pages.
- *Auxiliary Storage Manager* (ASM): Responsible for transferring virtual pages between real and auxiliary memory. When RSM detects a shortage of real storage, ASM frees these page slots by writing them to the page data set. If the page is referenced while it is not assigned a real storage slot, RSM calls ASM to read the page into central storage. ASM manages the transfer by initiating the I/O and by maintaining tables to reflect the current status of the auxiliary memory.

Paging and swapping

A virtual storage page can be situated in the real memory or in the auxiliary memory. z/OS transfers the page into real memory or out from the real memory. This movement of pages is called *paging*.

z/OS paging is transparent to the user. During job execution, only those pieces of the application that are required are brought in, or paged in, to central memory. To select pages for paging out to auxiliary memory, z/OS follows a least-recently-used algorithm.

Idle users or jobs are quiesced. Before z/OS 1.8 these users were swapped out, transferred completely to auxiliary storage. In z/OS 1.8 the quiesced user will be moved by an “in-real swap” to idle parts of storage. These pages might be paged out to auxiliary storage if they are needed by more active users.

3.3.2 Workload Manager (WLM)

One of the strengths of the System z platform and its operating system is the ability to run multiple workloads at the same time and manage the relative

priorities of these multiple workloads. The function that makes this possible is the Workload Manager component of z/OS.

The idea of z/OS Workload Manager (WLM) is to translate the business objectives associated with a given workload into technical constructs (rules and priorities) enforced by the operating system.

The definition of the workload items and the rules for managing them are gathered in the WLM Policy. Each workload is defined, and each item is assigned a goal and importance.

These goals define the expectation or service level agreement (SLA) of how the work should perform. WLM uses these goal definitions to manage the work in either a single system or a group of systems in a sysplex environment. WLM manages the use of system resources (such as processors, memory, and I/O devices) in order to honor those goals.

The identification and classification of work requests is supported by the middleware and the operating system: They inform WLM when a new unit of work enters the system or middleware and when it leaves the system or middleware. When the unit of works enters the system, it gets classified and assigned a service class that represents the performance goal that we want to achieve for this business unit and the importance of this work item relative to all the other work items. The performance goal objective stays with the business unit of work until it completes and WLM will manage the system resources to be able to achieve the desired goal.

Figure 3-1 shows the concept of WLM classification, the classification rules and the Service Class assignment.

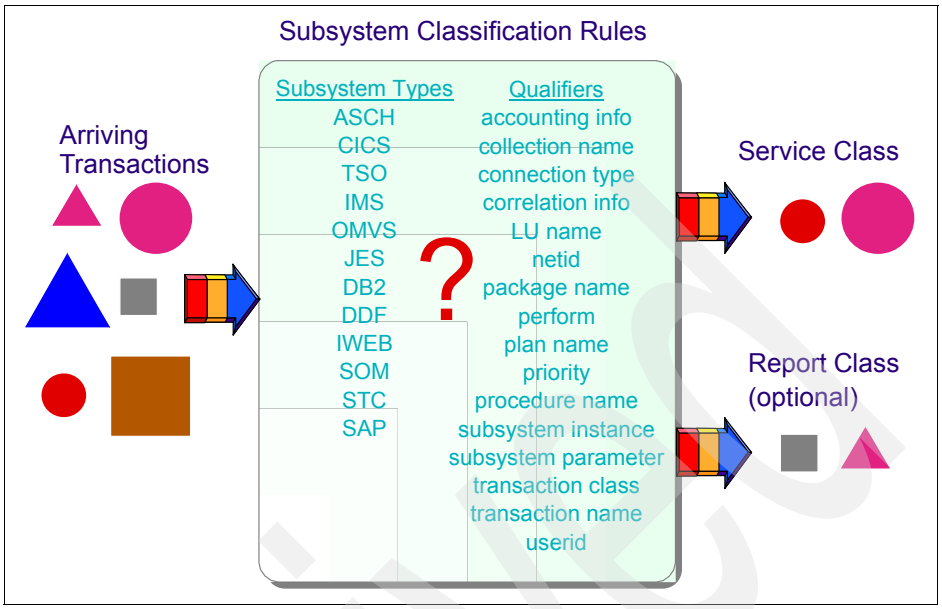


Figure 3-1 Classification of work requests

WLM has different types of performance goals that can be assigned. Certain types of goals are more appropriate for different types of workload.

Transactional workload is easily represented and measured with a concept of average response time. WLM monitors the transaction and calculates the average response time by measuring the time between the point when the work enters and leaves the system. This time is averaged for the transaction completions in an interval. Because a system deals with many work requests running in parallel, the time of an individual request becomes less important. The average time the work needs to fulfill its task on the system is more interesting because this is also the time the end user can observe while waiting for a response from the system.

Where response times for a given workload are distributed unevenly, the average response time can be influenced by very few long-running transactions. To become more independent from such events, WLM provides a different goal definition. For example, we would like a certain amount of work requests to end within a predefined time. Other work requests may run much longer, but we will consider the goal as achieved when the defined amount of requests ends in the expected time period. This goal is named a *percentile response time goal* and defines the percentage of all work requests that must end within a predefined time limit.

There are situations where it is not possible to capture the response times of the transactions: For example, a piece of middleware might not be instrumented to inform WLM about the transaction behavior, or transactions are too few and too sporadic. In these cases, or even for long-running tasks such as system address space, it might not make sense to manage them toward a response time goal.

Also, many processes provide services in the system, which cannot be easily attributed to a certain type of transaction. For those processes, it is also necessary to understand how fast they progress in the system and define a goal for them. Because the system is not able to capture the response time of this type of work, WLM uses a sampling mechanism to understand whether a work is suffering any constraints. This information can be obtained by observing where such delay situations might occur and continuously taking samples. These samples are used to identify how often individual work and the service classes were delayed for a resource and how often they were able to use it. The speed in the system can now be expressed by the acceptable amount of delay for the work when it executes. The speed is called *velocity*, a value from slowest to fastest, where fastest means that no measured samples show resource constraints and that the work did not encounter any delays for resources managed by WLM, and slowest means that the work was completely delayed over the measurement interval.

Certain transaction managers can create a new type of work called an *enclave*. As with other work items, this enclave is classified into a service class. The enclave has the special property that the work it represents can be part of multiple address spaces. After the service class has been assigned, it is kept with the enclave, and work from different server address spaces will be managed by the goal of the enclave. For example, a DB2 region can have many requestors getting data. Each of these requests runs in the DB2 address spaces at a priority appropriate for the goal of the requestor's service class. With an enclave requestor, the service class will follow the unit of work throughout the different server address spaces within the system or across systems within the same sysplex. The response time will be calculated when the enclave ends. This ensures that the unit of work maintains the same business goal independent of the middleware or system where it runs.

A number of benefits arise from WLM goal-oriented performance management. The most obvious benefit is the simplification in defining performance objectives and initialization states to the system. The installation is able to specify business objectives directly to the system in business terms. It is still the installation responsibility to ensure that each service class contains work with similar goals, business importance, and resource requirements in order to acquire the maximum benefit from WLM. With proper planning, the policy will still be appropriate even if the hardware platform is upgraded in size.

Sysplex workload distribution

WLM can help distribute workload across multiple system images in a sysplex. Sysplex distributor checks with WLM for a recommendation on where to route a new work request. WLM checks the workload on each system image and returns a list of eligible servers that can take care of the new work. The highest server on the list is the preferable server.

The objective is to send work where there is available capacity or, if there is no available capacity, where the least important work is running.

Transaction management

Client work requests span multiple address spaces on z/OS. Managing these requests requires recognizing the transaction flow and, based on this, managing the address spaces and requests to meet the end-user expectation.

External clients can be anything, for example a user sitting at a terminal or a program on a different server. The address space that receives the work request on z/OS primarily checks its authority and resource requirements. If the application can process it, the region sends it to an application processing region. The application region starts a program that processes the request, and the program invokes other middleware, operating system services, or services from other subsystems to complete the request.

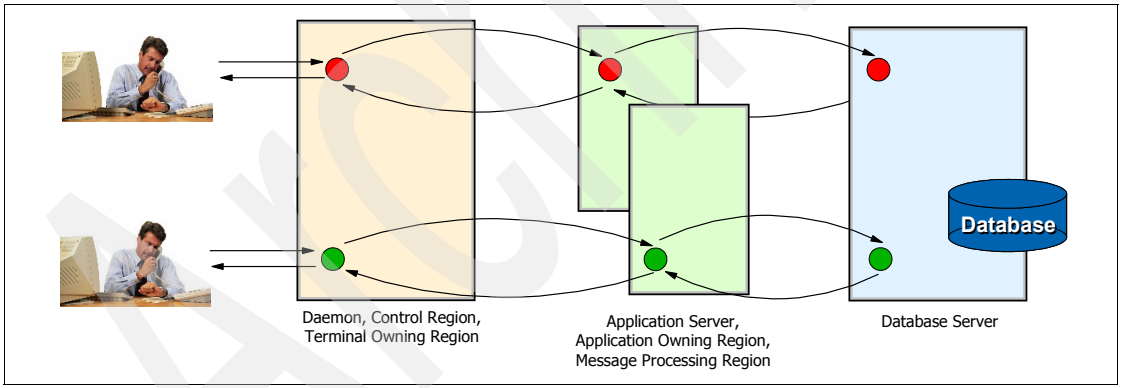


Figure 3-2 A transaction processing

For example, a CICS transaction that accesses a DB2 database will have tasks or threads that run in the CICS and the DB2 address spaces. Those tasks can be associated with a performance goal for the transaction, instead of with performance goals associated with the CICS and DB2 address spaces.

WLM has developed a set of programming interfaces that enable middleware and subsystems to identify work requests to WLM, help WLM to trace the work through the system, and identify the time the requests stay on z/OS.

3.3.3 Resource Recovery Services (RRS)

A transaction that accesses multiple databases and gets services from multiple server regions must be completely processed or completely backed-out if there was any error. For example, a transaction that moves money from one account to another must either complete or, in the case of any system failure, must be completely backed out. A debit must not be processed without the corresponding credit also being processed. This is true even if the two accounts are managed by different database subsystems.

z/OS provides the RRS subsystem to act as a the coordinator of transaction recovery. Transaction managers can call RRS services to coordinate the recovery of all services that the transaction uses. Before RRS was made available, the various transaction managers were required to write their own transaction recovery code and ensure that they could cooperate with every other database management system. RRS provides a global syncpoint manager that any transaction or database manager on z/OS can exploit. It enables transactions to update protected resources managed by many resource managers without having to write code to provide this function.

3.3.4 System Management Facility (SMF)

In a multitasking environment such as z/OS, it is important to collect information about the behavior of the system and the work that runs on that system. z/OS components and subsystems collect statistical data, and pass it to SMF to write in the SMF data set.

SMF system records include information about the configuration, paging activity, workloads, and resource usage. The records include information about CPU consumption, I/O activities, and data set activities for each of the task steps.

These SMF records can be used for capacity planning of system infrastructure elements as well as for accounting of customer usage of the system.

3.3.5 Systems Managed Storage

In a z/OS system, data management involves the allocation, placement, monitoring, migration, backup, recall, recovery, and deletion of data sets. These activities can be done either manually or through the use of automated processes. When data management is automated, the operating system

determines object placement and automatically manages data set backup, movement, space, and security. File data is stored in pools of disk devices according to the file attributes. Files that have not been used recently are migrated to a more cost-effective media, including tape. Critical data may be replicated to other storage devices automatically and continuously.

Depending on how a z/OS system and its storage devices are configured, a user or program can directly control many aspects of data set usage, and in the early days of the operating system, users were required to do so. Today, to reduce complexity and improve storage management tasks, z/OS installations rely on installation-specified settings in their DFSMS™ policy for data allocation and resource management.

DFSMS

Data Facility Storage Management Subsystem is an operating environment that helps automate and centralize the management of storage based on the policies that your installation defines for availability, performance, space, and security. DFSMS has multiple data management functional components as an integrated single software package:

DFSMSdfp™ Provides storage, data, program, and device management.

DFSMSdss™ Provides data movement, copy, backup, and DASD management functions.

DFSMShsm™ Provides backup, recovery, migration, and DASD space management functions.

DFSMSrmm™ Provides library management functions for removable media such as tape cartridges and optical media.

The storage management subsystem (SMS) is a DFSMSdfp facility designed for automating and centralizing storage management. SMS automatically assigns attributes to new data when that data is created. SMS automatically controls system storage and assigns data to the appropriate storage device.

The system programmer or storage administrator defines policies that automate the management of storage and hardware devices. These policies describe data allocation characteristics, performance and availability goals, backup and retention requirements, and storage requirements for the system.

The policies defined in your installation represent decisions about your resources. Those decisions have a direct impact on your IT cost efficiency and your IT availability. When making your policies, ask yourself some questions:

- What performance objectives are required by applications accessing the data? Based on these objectives, DFSMS will place data on devices that can provide the device response time you request. Data with high performance objectives

will be automatically placed on cache controllers, or striped over several devices and channels. By tracking data set I/O activity recorded in the SMF data, you can make better decisions about data set caching policies and improve overall system performance.

- When and how to back up data?
  Do certain data sets require incremental backups or total backups? You should determine the backup frequency, the number of backup versions, and the retention period by consulting user group representatives. This information is then used to define the backup policy for DFSMShsm. Be sure to consider whether certain groups of data sets need to be backed up in a consistent set.
- Whether data sets should be kept available for use during or after backup or copy?
  You can store backup data sets on disks or tape. Your choice depends on how fast the data must be recovered, media cost, operator cost, floor space, power requirements, air conditioning, the size of the data sets, and whether you want the data sets to be portable.
- How to manage backup copies kept for disaster recovery?
  Will the data be retained locally or in an off-site location, or both? Related data sets should be backed up in aggregated tapes. Each application should have its own, self-contained aggregate of data sets. If certain data sets are shared by two or more applications, you might want to ensure application independence for disaster recovery by backing up each application that shares the data. This information is used to build the DFSMS policy.
- What to do with data that is obsolete or seldom used?
  Data is obsolete when it exceeds its expiration dates and is no longer needed.

SMS can help convert those answers into IT actions, and thus help IT manage one of its resources in a better way.

3.3.6 Global resource serialization (GRS)

The z/OS environment can run many programs simultaneously. Some programs might require access to the same data set or resources at the same time. The operating system must provide a mechanism to allow this access without compromising the integrity of the target resource.

Resource serialization is the technique used to coordinate access to resources that are used by more than one application. Programs that change data need exclusive access to the data. Otherwise, if several programs were to update the same data at the same time, the data could be corrupted. (This is also referred to as a loss of data integrity). On the other hand, programs that only need to read data might be able to safely share access to the same data at the same time.

Global resource serialization (GRS) is the z/OS component that handles the serialization requests. GRS processes requests for resources from programs running both on a single z/OS image and in a sysplex of multiple z/OS images. GRS serializes access to resources to protect their integrity. When GRS grants shared access to a resource, exclusive users cannot obtain access to the resource. Likewise, when GRS grants exclusive access to a resource, all other requestors for the resource wait until the exclusive requestor frees the resource.

Enqueuing

Enqueuing is the means by which a program running on z/OS requests control of a serially reusable resource. Enqueuing is accomplished by using the ENQ (enqueue) and DEQ (dequeue) macros, which are available to all programs running on the system. For devices that are shared between multiple z/OS systems, enqueuing is accomplished through the RESERVE and DEQ macros.

On ENQ and RESERVE, a program specifies the names of one or more resources and requests shared or exclusive control of those resources. If the resources are to be modified, the program must request exclusive control; if the resources are not to be modified, the program *should* request shared control, which enables sharing of the resource by other programs that do not require exclusive control. If the resource is not available, the system suspends the requesting program until the resource becomes available. When the program no longer requires control of a resource, it uses the DEQ macro to release it.

Locking

Through locking, the system serializes the use of system resources by authorized routines. A lock is simply a named field in storage that indicates whether a resource is being used and who is using it. Locking is required in multiprocessor architectures to serialize the multiple processors' access to shared storage and other shared resources.

To use a resource that is protected by a lock, a routine must first request the lock for that resource. If the lock is unavailable (that is, it is already held by another program or processor), the action taken by the program or processor that requested the lock depends on whether the lock is a *spin lock* or a *suspend lock*:

- If a spin lock is unavailable, the requesting processor continues testing the lock until the other processor releases it. As soon as the lock is released, the requesting processor can obtain the lock and, thus, control of the protected resource.
- If a suspend lock is unavailable, the unit of work requesting the lock is delayed until the lock is available. Other work is dispatched on the requesting processor. All local locks are suspend locks.

You might wonder what would happen if two users each request a lock that is held by the other? Would they both wait forever for the other to release the lock first, in a kind of stalemate? In z/OS, such an occurrence would be known as a *deadlock*. Fortunately, the z/OS locking methodology prevents deadlocks.

To avoid deadlocks, locks are arranged in a hierarchy, and a processor or routine can unconditionally request only locks higher in the hierarchy than locks it holds.

3.4 Tools

This section lists management tools that are specific to z/OS.

3.4.1 Resource Measurement Facility (RMF)

RMF is an IBM product for performance analysis, capacity planning, and problem determination in a z/OS host environment. Many different activities are required to keep System z running smoothly, and to provide the best service on the basis of the available resources and workload requirements. This work is done by system operators, administrators, programmers, or performance analysts.

RMF can:

- Determine that a system is running smoothly.
- Detect system bottlenecks caused by resource contention.
- Evaluate the service an installation provides to different groups of users.
- Identify the delayed workload and the reason for the delay.
- Monitor system failures, system stalls, and failures of selected applications.

RMF provides its benefits through the operation of Postprocessing and Online Monitoring functions. They are based on a set of data gatherers and data services that enables access to all performance-relevant data in a z/OS environment. RMF has three data gatherers/reporters known as monitors:

Monitor I Provides long-term data collection for system workload and system resource utilization. The Monitor I session is continuous, and measures various areas of system activity over a long period of time

Monitor II Provides online measurements on demand for solving immediate problems. A Monitor II session can be regarded as a snapshot session collecting data about address space states and resource use. Both Monitor I and Monitor II can write the data they gather to SMF (see “System Management Facility (SMF)” on page 46).

Monitor III Provides short-term data collection and online reports for continuous monitoring of system status and for solving performance problems, workflow delay monitoring, and goal attainment supervision.

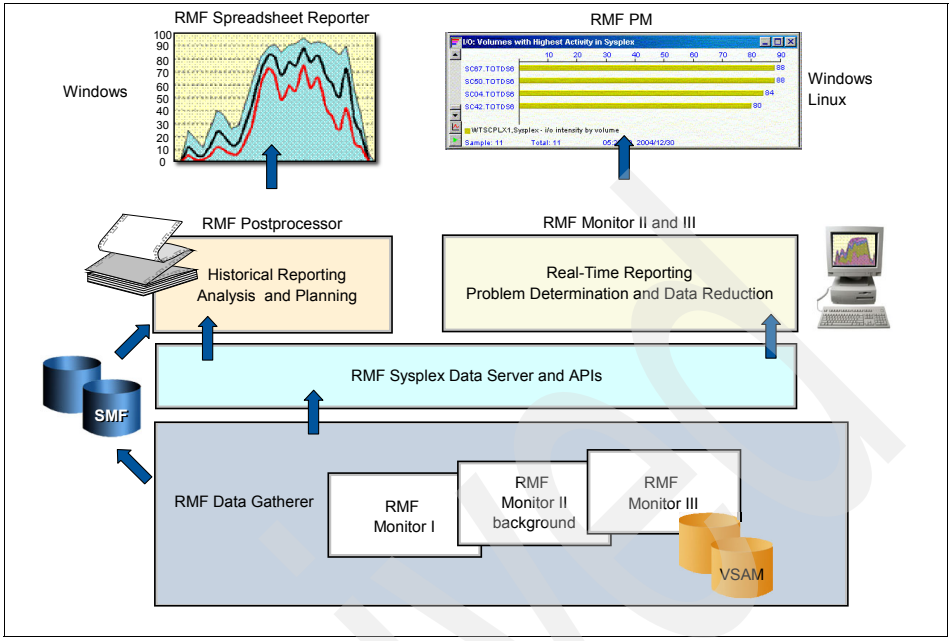


Figure 3-3 RMF product overview

RMF Performance Monitor

RMF PM, a Java-based monitor, enables you to monitor the performance of your z/OS system from a single workstation. RMF PM takes its input data from a single data server on one system in the sysplex, which gathers the data from the RMF Monitor III on each MVS image. Figure 3-4 is an example of one of the capabilities you have on RMF PM.

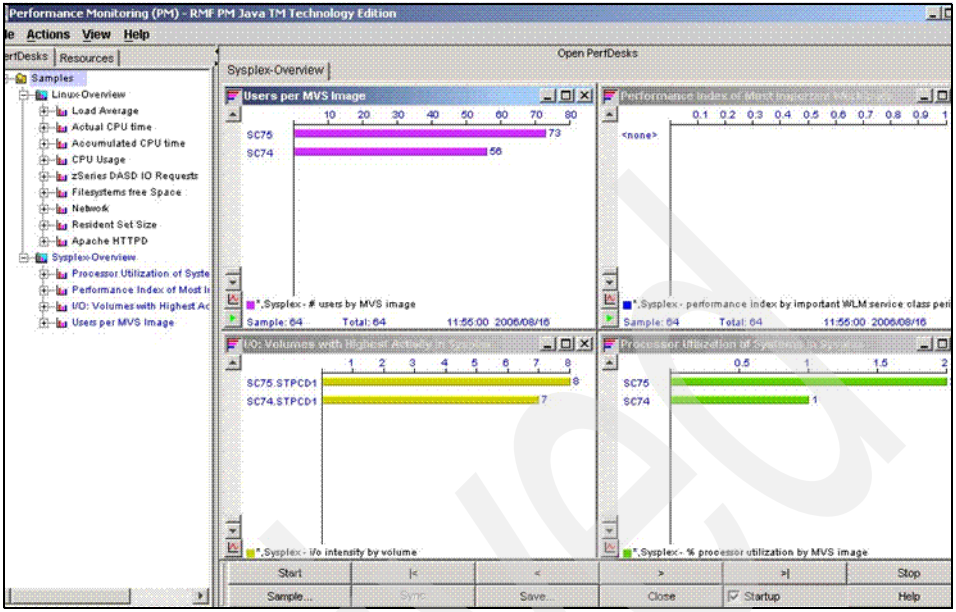


Figure 3-4 RMF PM

Note: Resource Measurement Facility (RMF) is an optional product that ships with every release of z/OS at the current level of support.

3.4.2 IBM OMEGAMON z/OS Management Console

The IBM OMEGAMON® z/OS Management Console is a new, no-charge monitoring product designed to help the new generation of IT professionals. The console's advanced graphical user interface (GUI) delivers real-time health-check information (provided by the IBM Health Checker for z/OS) and configuration status information for z/OS systems and sysplex resources.

The IBM OMEGAMON z/OS Management Console has built-in alerting and expert advice capabilities that can offer detailed contextual information about alerts and corrective actions.

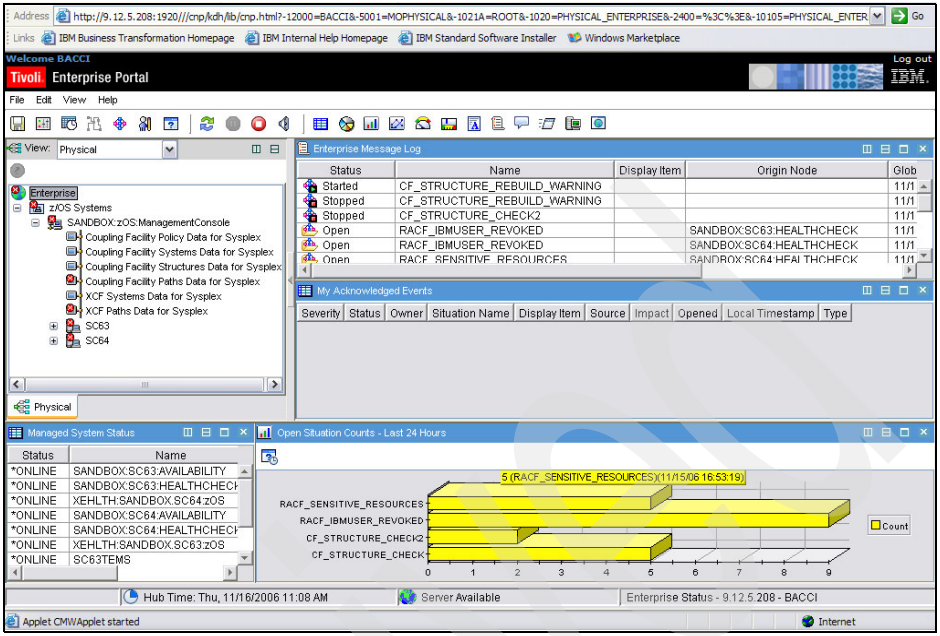


Figure 3-5 z/OS management console

3.4.3 SMP/E

The z/OS environment is required to be available 24x7. Upgrading, fixing, and installing software should be done quickly and accurately. As with other OSs, z/OS has many components and many products (IBM and non-IBM) running on top of the operating system. The system programmer has a number of tasks to perform. Installing a new product, or updating an existing product, the system programmer has to ensure that all prerequisites of the product are installed. Fixes to the products have to be researched, installed in a test environment, and eventually, after verification, be deployed to the production environment.

Tracking all of the modules and fixes to the modules in a z/OS system is an extremely complicated process. It cannot be done manually or with simple automated software. SMP/E is the z/OS tool for managing the installation of software products on a z/OS system and for tracking modifications to those products. SMP/E controls these changes at the component level by:

- Selecting and proper levels of code to be installed from a large number of potential changes.
- Calling system utility programs to install the changes.
- Keeping records of the installed changes by providing a facility to enable you to inquire on the status of your software and to reverse the change if needed.

All product code, modifications to the code, and the inventory of the installed code are located in the SMP/E database.

3.4.4 Programming languages

System z platform offers all tools needed for implementing industry-standard software engineering methodologies.

Language environment

IBM Language Environment® for z/OS & z/VM (Language Environment) provides a single run-time environment for C, C++, COBOL, Fortran, PL/I. and assembler applications. The Language Environment common library includes common services such as messages, date and time functions, math functions, application utilities, system services, and subsystem support. All of these services are available through a set of interfaces that are consistent across programming languages. You may either call these interfaces yourself or use language-specific services that call the interfaces. All of this provides consistent and predictable results for your applications, independent of the language they are written in.

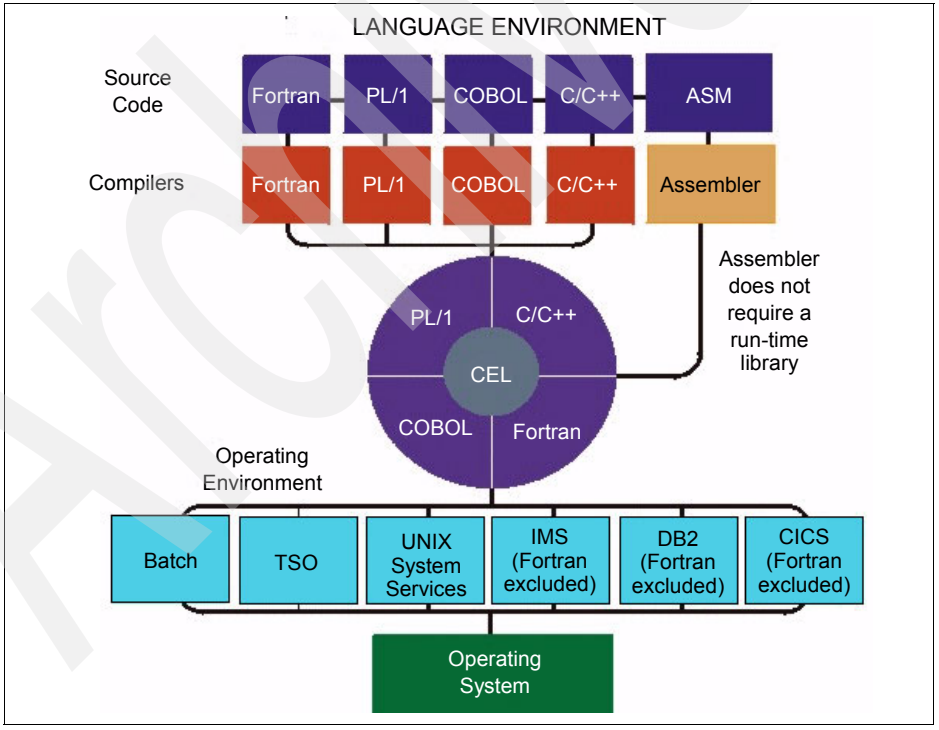


Figure 3-6 Language Environment

Java

Java is an object-oriented programming language developed by Sun™ Microsystems Inc. Java can be used for developing traditional mainframe commercial applications as well as Internet and intranet applications that use standard interfaces.

Java is an increasingly popular programming language used for many applications across multiple operating systems. IBM is a major supporter and user of Java across all of the IBM computing platforms, including z/OS. The z/OS Java products provide the same, full-function Java APIs as on all other IBM platforms. In addition, the z/OS Java licensed programs have been enhanced to allow Java access to z/OS unique file systems. Programming languages such as Enterprise COBOL and Enterprise PL/I in z/OS provide interfaces to programs written in Java.

The various Java Software Development Kit (SDK) licensed programs for z/OS help application developers use the Java APIs for z/OS, write or run applications across multiple platforms, or use Java to access data that resides on the mainframe. Some of these products allow Java applications to run in only a 31-bit addressing environment. However, with 64-bit SDKs for z/OS, pure Java applications that were previously storage-constrained by 31-bit addressing can execute in a 64-bit environment. Also, the System z9 processors support a special CP for running Java applications called the System z Application Assist Processor (zAAP). Using a zAAP engine adds capacity to the platform without increasing software charges. Java programs can be run interactively through z/OS UNIX® or in batch.

Traditional business application languages

(COBOL, PL/1, Assembler, C/C++)

The Java platform offers many attractive characteristics for building modern software systems. Programmers already experienced with object-oriented languages typically find Java relatively easy to learn and use. But developers familiar with procedural programming, fourth-generation languages (4GLs), and other traditional development technologies often find Java complex—so much so that they resist opportunities to use it. They instead continue developing with the programming technologies with which they are most comfortable.

Enterprise Generation Language (EGL) is specifically designed to help the traditional developer leverage all of the benefits of Java and COBOL, yet avoid learning all of its details. EGL is a simplified high-level programming language that enables you to quickly write full-function applications based on Java and modern Web technologies. For example, developers write their business logic in EGL source code, and from there, the EGL tools generate Java or COBOL code,

along with all runtime artifacts needed to deploy the application to the desired execution platform.

EGL hides the details of the Java and COBOL platform and associated middleware programming mechanisms. This frees developers to focus on the business problem rather than on the underlying implementation technologies. Developers who have little or no experience with Java and Web technologies can use EGL to create enterprise-class applications quickly and easily.

IBM Rational® COBOL Generation Extension for zSeries provides the ability to continue reaping the benefits of the highly scalable, 24x7 availability of the System z platform by enabling procedural business developers to write full-function applications quickly while focusing on the business aspect and logic and not the underlying technology, infrastructure, or platform plumbing.

Built on open standards, Rational COBOL Generation for zSeries adds valuable enhancements to the IBM Software Development Platform so you can:

- Provide an alternative path to COBOL adoption.
- Construct first-class services for the creation and consumption of Web Services for service-oriented architecture.
- Hide middleware and runtime complexities.
- Achieve the highest levels of productivity.
- Migrate from existing technologies to a modern development platform.
- Deliver applications based on industry standards that interoperate with existing systems.
- Easily retrain procedural business programmers to be highly productive in the Java 2 Enterprise Edition (J2EE) world.
- Exploit visual programming techniques for Web development and code automation capabilities for rapid development of application business logic.

3.4.5 Software development tools

IBM WebSphere Developer for System z V7 includes capabilities that can help make traditional mainframe development, Web development, and integrated SOA-based composite development fast and efficient. COBOL, PL/I, C, C++, High-Level Assembler, and Java developer communities can also be more productive when they take advantage of these functions.

WebSphere Developer for System z V7.0 supports industry standards that can simplify the development of rich client interfaces, Web user interfaces, traditional terminal interfaces, and back-end, business logic deployable to CICS, IMS, DB2, Batch, UNIX System Services, and WebSphere transactional environments. It

also includes high-productivity tools for business-oriented developers, new to Java 2 Enterprise Edition (J2EE) and Java, that integrate IBM eServer System z processing via Web services with powerful Web User and Session management.

IBM WebSphere Developer for zSeries is optimized for IBM WebSphere software and System z deployment environments, and supports multi-vendor runtime environments. It is powered by the Eclipse open source platform so developers can adapt and extend their development environment to match their needs and increase their productivity.

With WebSphere Developer for zSeries, you can:

- Accelerate the development of:
  - Service-oriented architecture (SOA) including services deployable to CICS, IMS, WebSphere Application Server, and DB2 Stored Procedure environments
  - Dynamic Web applications including Java and J2EE
  - Traditional COBOL and PL/I applications, C, C++, and High-Level Assembler applications, and Java applications that are created for and run under JZOS on z/OS
  - Web services to integrate these applications
- Deploy to multiple run times including WebSphere, CICS, IMS, batch, and DB2 via Stored Procedures.
- Leverage existing skills to write Web or COBOL applications by using the high-level Enterprise Generation Language.
- Adapt and extend the development environment with a wide range of plug-ins.
- Visualize and graphically edit J2EE code through the UML Visual Editor.
- Detect performance issues early with performance profiling and trace tools for WebSphere applications.
- Generate Enterprise COBOL XML adapters for CICS and IMS Web service-based applications.
- Generate WSDL and JavaBeans™ for testing and deploying Web services.

3.5 Middleware

In this section, we talk about middleware that is available for the System z platform that are used to support online or interactive workloads, such as transaction processing, database management, and Web serving.

3.5.1 Business layer

Transactions occur in everyday life, for example, when you exchange money for goods and services or do a search on the Internet. A transaction is an exchange, usually a request and response, that occurs as a routine event in running the day-to-day operations of an organization.

Transactions have the following characteristics:

- Small amount of data is processed and transferred per transaction
- Large numbers of users
- Executed in large numbers, often at high rates

A business transaction is a self-contained business deal. Some transactions involve a short conversation, and others involve multiple actions that take place over an extended period.

A single transaction might consist of many application programs that carry out the processing needed. Large-scale transaction systems rely on the multitasking and multithreading capabilities of z/OS to allow more than one task to be processed at the same time, with each task saving its specific variable data and keeping track of the instructions each user is executing.

Transaction systems must be able to support a large number of concurrent users and transaction types. One of the main characteristics of a transaction or online system is that the interactions between the user and the system are very brief. Most transactions are executed in short time periods (one second or less, in some cases). The user will perform a complete business transaction through short interactions, with immediate response time required for each interaction. These are mission-critical applications; therefore, continuous availability, high performance, and data protection and integrity are required.

Transaction processing requires:

- Immediate response time
- Continuous availability of the transaction interface to the end user
- Security
- Data integrity

In fact, a transactional system has many of the characteristics of an operating system:

- Managing and dispatching tasks
- Controlling user access authority to system resources
- Managing the use of memory
- Managing and controlling simultaneous access to data files
- Providing device independence

In a transaction system, transactions must comply with four primary requirements known jointly by the mnemonic A-C-I-D or ACID:

Atomicity The processes performed by the transaction are done as a whole or not at all. For example, when moving money from one account to another the debit and the credit must both be either completed or both backed out. The data cannot be half-processed.

Consistency The transaction must work only with consistent information. The database must be in a legal state when the transaction begins and when it ends.

Isolation The processes coming from two or more transactions must be isolated from one another. This means that no operation outside the transaction can ever see the data in an intermediate state.

Durability The changes made by the transaction must be permanent. When the user has been notified of success, the transaction will persist, and not be undone. This means it will survive any failure or combinations of failures in the system or subsystems.

Usually, transactions are initiated by an end user who interacts with the transaction system through a terminal. They can be submitted in any of the following ways:

- Web page
- Remote workstation program
- Application in another transaction system
- Triggered automatically at a predefined time

An important concept in transactional systems is the *commit* and *roll back*. Commit and roll back refer to the set of actions used to ensure that an application program either makes *all* changes to the resources represented by a single unit of recovery (UR), or makes *no* changes at all. The two-phase commit protocol provides commit and rollback. It verifies that either all changes or no changes are applied even if one of the elements (such as the application, the system, or the resource manager) fails. The protocol allows for restart and recovery processing to take place after system or subsystem failure. RRS, described in 3.3.3, “Resource Recovery Services (RRS)” on page 46 is the z/OS subsystem that provides these services.

The most common transactional middleware for the z platform are CICS/TS, IMS/TM, and WebSphere Application Server. The following sections provide a brief description of these major transactional systems.

CICS Transaction Server

CICS provides a layer of function for managing transactions, removing the need to handle the infrastructure within the business logic. For example, when an application program accesses a terminal or any device it doesn’t communicate

directly with it. The program issues commands to communicate with CICS, which communicates with the needed access methods of the operating system. Finally, the access method communicates with the terminal or device.

A z/OS system might have multiple copies of CICS running at one time. Each CICS starts as a separate z/OS address space. CICS provides an option called multi-region operation (MRO), which enables the separation of different CICS functions into different CICS regions (address spaces). So a specific CICS address space (or more) might do the terminal control and will be named terminal owning region (TOR); another CICS region might host the application layer, the so-called application-owning regions (AORs).

CICS enables you to keep your application logic separate from your application resources.

You can use COBOL, OO COBOL, C, C++, Java, PL/I, or Assembler language to write CICS application programs to run on z/OS. Most of the processing logic is expressed in standard language statements, but you use CICS commands, or the Java and C++ class libraries, to request CICS services.

IMS/TM

Created in 1969 as Information Management System/360, IMS is both a transaction manager and a database manager for z/OS. IMS consists of three components: the Transaction Manager (TM), the Database Manager (DB), and a set of system services that provide common services to the other two components.

As IMS developed over the years, new interfaces were added to meet new business requirements. It is now possible to access IMS resources using a number of interfaces to the IMS components.

In this section, we provide a brief description of the transaction manager functions of IMS; we discuss the database functions more thoroughly in “IMS Database Manager” on page 64.

You write an IMS program in much the same way you write any other program. You can use COBOL, OO COBOL, C, C++, Java, PL/I, or Assembler language to write IMS application programs.

The IMS Transaction Manager provides users of a network with access to applications running under IMS. The users can be people at terminals or workstations, or they can be other application programs either on the same z/OS system, on other z/OS systems, or on non-z/OS platforms.

A transaction is a setup of input data that triggers the execution of a specific business application program. The message is destined for an application program, and the return of any results is considered one transaction.

An IMS subsystem runs in several address spaces in a z/OS system. There is one controlling address space and several dependent address spaces providing IMS services and running IMS application programs.

To make the best use of the unique strengths of z/OS, IMS has these features:

- Runs in multiple address spaces. IMS subsystems (except for IMS/DB batch applications and utilities) normally consist of a control region address space, dependent address spaces providing system services, and dependent address spaces for application programs.
- Runs multiple tasks in each address space. IMS, particularly in the control regions, creates multiple z/OS subtasks for the various functions to be performed. This allows other IMS subtasks to be dispatched by z/OS while one IMS subtask is waiting for system services.
- Uses z/OS cross-memory services to communicate between the various address spaces that make up an IMS subsystem. It also uses the z/OS Common System Area (CSA) to store IMS control blocks that are frequently accessed by the IMS address spaces, thus minimizing the overhead of using multiple address spaces.
- Uses the z/OS subsystem feature. IMS dynamically registers itself as a z/OS subsystem. It uses this facility to detect when dependent address spaces fail and prevent cancellation of dependent address spaces (and to interact with other subsystems such as DB2 and WebSphere MQ).
- Can make use of a z/OS sysplex. Multiple IMS subsystems can run on the z/OS systems that make up the sysplex and can access the same IMS databases.

WebSphere Application Server

As enterprises move many of their applications to the Web, mainframe organizations face the complexity of enabling and managing new Web-based workloads in addition to more traditional workloads, such as batch processing. WebSphere Application Server is a comprehensive, sophisticated J2EE and Web services technology-based application system and e-business application deployment environment. WebSphere Application Server on z/OS is the J2EE implementation conforming to the current software development kit (SDK) specification supporting applications at an API level.

This Java application deployment and run-time environment is built on open standards–based technology such as CORBA, HTML, HTTP, IIOP, and J2EE-compliant Java technology standards for servlets, JavaServer™ Pages

(JSP™) technology, and Enterprise JavaBeans™ (EJB™), and it supports all Java APIs needed for J2EE compliance.

The application server runtime is highly integrated with all inherent features and services offered on z/OS. The application server can interact with all major subsystems on the operating system including DB2, CICS, and IMS. It has extensive attributes for security, performance, scalability and recovery. The application server also uses sophisticated administration and tooling functions, thus providing seamless integration into any data center or server environment.

The Controller Address Space will automatically start a servant region as work arrives. An application server instance is composed of a controller region (CR) and one or more servant regions (SRs).The application server on z/OS supports two types of configurations: Base and Network Deployment. Each configuration uses essentially the same architectural hierarchy, comprised of *servers*, *nodes,* and *cells*. However, cells and nodes play an important role only in the Network Deployment configuration.

WebSphere MQ

In an online application environment, messaging and queueing enables communication between applications on different platforms. IBM WebSphere MQ for z/OS is an example of software that manages messaging and queuing in the mainframe and other environments. With messaging, programs communicate through messages, rather than by calling each other directly. With queuing, messages are retained on queues in storage, so that programs can run independent of each other (asynchronously).

Here are some of the functional benefits of WebSphere MQ:

- A common application programming interface, the MQI, which is consistent across the supported platforms.
- Data transfer data with assured delivery. Messages are not lost, even if a system fails, nor is there duplicate delivery of messages.
- Asynchronous communication. That is, communicating applications need not be active at the same time.
- Message-driven processing as a style of application design. An application is divided into discrete functional modules that can run on different systems, be scheduled at different times, or act in parallel.
- Application programming is made faster when the programmer is shielded from the complexities of the network.

In addition to the standard capabilities, WebSphere MQ provides high availability features based on the sysplex that in many cases can be the decisive criterion for running the Queue Manager and related applications on z/OS.

The basic approach for a z/OS sysplex configuration to provide high availability is the idea that within the sysplex, there are multiple components with equivalent functionality. If one of the components fails, its duplicate component will take over the function with little or no impact to applications and users.

High availability for queue-based processes of all kinds are provided by the WebSphere MQ *Queue-sharing group* function, also known as *Shared Queues*. In order to provide full sysplex support with mutual backup functionality at any time, these specific queues are located in a central location (coupling facility) and accessible to a set of queue managers with shared access to the queue. The queue-sharing group concept provides a single system image, of the queue managers that make up a queue-sharing group, to the outside world and provides the redundancy necessary to provide continuous operations.

3.5.2 Data layer

Several types of databases can be used on a mainframe to exploit z/OS: inverted list, hierarchic, network, or relational. Mainframe sites tend to use a hierarchical model when the data *structure* (not data values) of the data needed for an application is relatively static.

Both database systems are capable of reducing the application programming effort; providing a more efficient way to create, modify and access data; and providing a greater level of data security and confidentiality than a flat file system. However, a relational database has the additional, significant advantage over the hierarchical type of being non-navigational. By *navigational*, we mean that in a hierarchical database, the application programmer must know the structure of the database. The program must contain specific logic to navigate from the root segment to the desired child segments containing the desired attributes or elements. The program must still access the intervening segments, even though they are not needed.

The following section provides a brief description of the IBM database managers.

DB2 for z/OS

DB2 for z/OS is the IBM relational database solution on the System z platform. It was one of the early subsystems on this platform to add support for Parallel Sysplex data sharing and has been at the forefront of Parallel Sysplex exploitation for more than 10 years.

DB2 for z/OS has the peculiarity to implement a shared cache cluster model. Multiple DB2 subsystems running on multiple z/OS systems across multiple logical partitions are presented to applications as one single database instance. Applications can access any member of the DB2 data sharing group with full read/write access to all data.

The DB2 data sharing group uses caching in the coupling facility to share the data.

IMS Database Manager

The IMS Database Manager provides a central point for the control of and access to application data. IMS provides a full set of utility programs to provide all of these functions within the IMS product.

IMS uses a hierarchical model as the basic method for storing data, which is a pragmatic way of storing the data and implementing the relationships between the various types of entities. In this model, the individual entity types are implemented as segments in a hierarchical structure. The hierarchical structure is determined by the designer of the database, based on the relationships between the entities and the access paths required by the applications.

Note: In the IMS program product itself, the term *database* is used slightly differently from its use in other DBMSs. In IMS, database is commonly used to describe the implementation of one hierarchy, so that an application would normally access a large number of IMS databases. Compared to the relational model, an IMS database is approximately equivalent to a table or set of tables.

4

Security

This chapter describes the security functionalities that are available on the System z platform, as a combination of hardware and software (z/OS) functions. For structural and packaging reasons, they are grouped as follows:

- z/OS Security Server
- z/OS Integrated Security Services
- z/OS cryptographic services
- System z hardware cryptographic functions
- IBM Encryption Facility for z/OS
- Public key infrastructure and Digital Certificate support

The chapter ends with a short listing of the security certifications achieved for System z hardware and its operating systems.

4.1 z/OS Security Server

It is generally believed that about 70 percent of all large enterprise business data resides on mainframe server platforms running IBM z/OS or its predecessor, OS/390®. From a security perspective, mainframe computer operating systems are the de facto intellectual standard to which newer operating systems are compared.

z/OS, from its beginnings as Multiple Virtual Storage (MVS) in the 1970s, was built on System/360 (S/360), and now System z, a strong hardware architecture that provides for the safe execution of multiple applications on one system.

Working storage for each application is separated from the storage of other applications, as well as from the storage used by the supervisor functions of the operating system. In addition, applications run in a different hardware “state” than supervisor functions and, therefore, do not have direct access to privileged hardware instructions. On top of this is layered a common point of user authentication and access control, both as a way to reduce the need for each application to provide these services uniquely, and as a way to reduce the administrative burden.

Twenty years ago, of course, this security structure needed only to support an environment of isolated batch processing systems or systems with terminal access through private corporate networks. Over time, and especially in the past decade, customer businesses have changed dramatically, reaching out to their suppliers, distributors, and customers through a variety of networking techniques. Businesses are now truly worldwide operations and rife with merger activity. In short, the nature of business has changed and so have the security requirements.

One change is that a single application, which can now start on the Internet, is expected to run across multiple platforms. As this has happened, OS/390 has further transformed into z/OS, and IBM has had to marry the security infrastructure with many open security standards.

A second change is a rise in the importance of encryption: IBM enterprise servers have a long history with encryption, having introduced optional hardware-based encryption in 1991. Since 1997, hardware encryption has been standard in S/390® processors and (now) System z processors. This is because IBM understands that encryption is the basis of security when using the Internet, and that customers require the performance and security that the IBM tamper-resistant hardware encryption provides.

Some security functions are provided in the base z/OS product, and others are packaged in an optional Security Server feature. Chief among the functions

packaged within the Security Server is the Resource Access Control Facility (RACF). RACF incorporates various elements of security, such as user identification, and authentication and access control.

The functions of the z/OS Security Server can be classified as follows:

- User identification and authentication
  - Basic authentication
  - Trusted third-party identification and authentication
- System Authorization Facility
- Access control
- Auditing and logging
- Networking and communications security

4.1.1 User identification and authentication

Identification and authentication are defined as the act of identifying yourself to the computing system and then proving that you are who you claim to be. Users identify themselves to a computing system application with a user ID and then prove that they are who they claim to be by correctly entering the password that has been associated (previously) with that user ID.

Identification and authentication technology, in one form or another, is implemented in several components of z/OS, using multiple security technologies. One of these forms is “basic authentication,” which depends on passwords for authenticating users.

Basic authentication

Instead of authorizing access to individual files by a password as in older operating systems, with RACF, the password authenticates the user’s authority to claim an identity. This technique is called *basic authentication*.

Authentication occurs when the user accesses the computing system and claims to be a particular user by specifying a user ID. The Security Server then uses the user ID as a search key to look up the predefined identity record associated with the claimed user ID within the user registry (the RACF database). If a record for the claimed user ID is found, the password that has also been stored in the record is compared with the password supplied by the entering user, and if the two match, the user is considered to be authenticated.

The authentication process results in the building of a piece of data whose generic name is a *credential* and which in z/OS is more precisely called an

Access Control Environment Element (ACEE). It is used to identify the authenticated user during access control authorization checking and for auditing.

For security reasons the password is encrypted with a one-way encryption algorithm.

Although the password is more secure than if it were stored in clear text in the registry, it is still subject to a “brute-force” hacker attack if the hacker is allowed to try many possible passwords until the correct one is found. On centralized systems, where the physical security of the registry is reliable, this form of attack is defeated by allowing only a small number of invalid attempts to be made against an individual user ID, within a reasonable time period, then revoking the user ID.

One potential problem of using the user ID and password (basic authentication) is that the password has to be transmitted from the user location to the location where the security server resides, and the security of the password while being transmitted is exposed.

With z/OS, there are several ways to avoid this exposure:

- Encrypt the password while it is transmitted. This can be achieved by using:
  - The Secure Sockets Layer (SSL) protocol
  - A Virtual Private Network (VPN) TCP/IP-encrypted tunnel
- Use a substitute to the password, called a *PassTicket*. PassTickets are generated dynamically by a trusted server, based on a secret encryption key that is shared with the target RACF-secured server. They can also be used to forward an authentication from one trusted server to another.

Trusted third-party identification and authentication

Although basic authentication is still largely used, it has some limitations when used with Internet applications. For example, basic authentication is convenient when there are only a few thousands user IDs to store and manage, but when there are millions of user IDs to manage, such as with Internet applications, basic authentication would require huge amounts of external memory (disk space) and most likely would create performance problems.

For these reasons, basic authentication is giving way to “trusted third-party” identification and authentication techniques. Modern trusted third-party identification and authentication techniques that are in common use include:

- The use of X.509 version 3 digital certificates with an associated public key infrastructure (PKI)
- The use of strong authentication through Kerberos

Digital certificates and PKI

A digital certificate can be used to identify and authenticate one user to another user (or server) and as the basis for generation of cryptographic keys for secure communication between these parties. Digital certificates that are created according to the popular X.509 v3 Internet standard are based on what is known as *public key cryptography*.

Public key cryptography uses two keys, a *key pair*, which are mathematically related to each other such that data enciphered with either key of the pair can only be deciphered with the other key of that pair. This basic technology can be used both to provide the foundation for private communication between parties and for the parties to be identified and authenticated to each other.

The Security Server for z/OS supports X.509 v3 digital certificates by allowing applications (the IBM HTTP server for z/OS, for example) to pass authenticated X.509 v3 digital certificates into RACF, then have RACF translate them using a mapping process into appropriate RACF identities.

The z/OS Security Server also supports a public key infrastructure (PKI) in the form of an end-user Web-initiated administrative process. Its function is the creation and automated distribution of user (and server) X.509 v3 digital certificates in large numbers, providing security appropriate to the needs of e-business.

Secure Sockets Layer

SSL is a public key cryptography-based extension to the TCP/IP “socket” interface that can be implemented at the application level. SSL has several functions that can be used by an e-commerce application to communicate with large numbers of users via common Internet browser software:

- Identification and authentication (with a high degree of trust) of server applications to users of the application.
- Establishment of a private cryptographic communication channel through the Internet between the communicating parties, such as between the customer on the Internet and the Web application server.
- Identification and authentication (with a high degree of trust) of application users to Web application servers. This function is most useful when the application provides function that requires the user to be known to the application.

SSL is widely supported in browsers and most Web servers. The first two functions are in common use worldwide, and the third (user identification and authentication using X.509 v3 digital certificates) is being applied in an increasing number of cases.

SSL represents the single most important user of cryptography in the spectrum of secure e-business applications.

Over the past few years IBM has focused on encryption performance, especially in support of SSL, and has added a performance-optimized system SSL service in z/OS that utilizes System z integrated cryptographic hardware to handle complex cryptographic operations.

As a result IBM has increased z/OS SSL performance 461-fold (from 13 SSL session handshakes per second in 1998 to 6000 per second in 2006).

Kerberos on z/OS

Another form of trusted third-party user identification and authentication is Kerberos. The fundamental difference between Kerberos and trusted third-party identification and authentication with digital certificates is the method of encryption used.

As previously discussed, X.509 v3 digital certificates technology is based on public key technology, more specifically on asymmetric cryptography, in which two mathematically related keys are used.

Kerberos, instead, is based on *symmetric* cryptography, in which the same key is used to both encipher and decipher information. Therefore, before being able to communicate securely with any server, the user must first contact the Kerberos server, be authenticated, and be passed a cryptographic key with which to speak to the target server.

Recent enhancements to the Security Server for z/OS include an implementation of the Kerberos user identification and authentication functions.

A unique feature of the z/OS Security Server Kerberos support is that the required Kerberos registry of users, along with the user passwords, is built on top of the existing RACF registry of users (for example, the RACF database).

That is, the required Kerberos information for each user is added to the RACF information for existing users, so an entirely new registry (another list of the same users) is not required, and administrators do not have to learn the administration user interface of another system. In the same vein, a user's Kerberos and RACF passwords are the same, so there are no password synchronization problems.

4.1.2 System Authorization Facility

Approximately 30 years ago IBM developed the Resource Access Control Facility (RACF) to provide centralized security functions such as user identification and authentication, resource access control, and auditing for both the operating

system and applications running on the system. RACF initially provided these functions through the operating system's "supervisor call" facility, enabling programs written in basic assembler language to invoke security functions via assembler macros. RACF existed as a separate product that customers could install as an option.

About 20 years ago, IBM implemented the System Authorization Facility (SAF) within the MVS operating system. SAF combined the various security function invocations into a single extensible security mechanism using a new macro instruction named RACROUTE, which provided subfunctions for each of the previously separate assembler language security macro instructions.

This new security structure provided several benefits, as follows:

1. SAF made it easier for application developers to find information about available security functions. Instead of trying to locate information on several separate security macro instructions, the application developer needed to look at information about only one instruction.
2. SAF made it easier for an application developer to interpret the results of a security operation. Rather than needing to handle many different results from many different security functions, with SAF and RACROUTE an application usually needed to handle only three consistent results from any security request.
3. SAF and RACROUTE made it easier to design applications. Applications no longer needed to check for the presence of a security product before making a request for security services, nor did applications need as much code to operate correctly with different, possibly somewhat incompatible, levels of the operating system or security product.

SAF also provided:

- Direct benefits for customers by providing a central mechanism (an exit point or hook within SAF itself) for customized security-request processing.
- Indirect benefits for customers, because security management became simpler as more applications used centralized security services rather than provide their own security mechanisms.

SAF may satisfy some security requests by itself or handle some jointly with the installed security product, although for most functions SAF merely routes control to the security product to provide the security services needed by applications.

This routing enables customers to provide an exit routine that may examine or modify an application's security request, and then, as needed, route the request to RACF or another installed security product.

4.1.3 Access control

Access Control is the process of determining who has access to what.

Who is determined by the identification and authentication, and *what* is determined by information on protected resources, including access rules.

When a user attempts to access a protected resource, a data file for example, the system component (resource manager) that supports the use of the resource invokes the system security manager (for example, RACF), requesting an access control decision. The security manager uses information about the user and the protected resource to determine whether the user is authorized to access the resource, and returns the decision to the requesting resource manager, which allows or denies access to the resource.

In making the access control decision, the security manager also considers additional factors, such as how the user intends to use the resource: for example, to read the information or to modify (update) it.

In most security managers, access rules are stored in entities called Access Control Lists (ACLs). ACLs are usually stored in the security manager data repository. In the case of RACF, ACLs are called *profiles* and are stored in the RACF database.

RACF also supports the concept of user groups, which are logical entities that represent several users with some common characteristics. The GROUP concept in RACF can greatly reduce the burden of administration of ACLs because an ACL can be associated once with a GROUP (of users) and have the same effect as if it had been associated individually with each user in the group.

Networking and communications security on System z servers is provided by the Communications Server component of z/OS.

4.1.4 Multilevel security

Another fundamental aspect of access control technology that is supported by RACF is the possibility to make access control decisions based on the compartmentalization of computer resources and information.

Compartmentalization is sometimes referred to as *multilevel security* (MLS).

The primary arena where MLS is valuable is government agencies that need a security environment that keeps information classified and compartmentalized between users. In addition to the fundamental identification and authentication of users, auditing and accountability of the actions by authenticated users on these systems is provided by the security environment.

Normally in such highly secure environments, to manage the compartmentalization of information between users, each compartment is on its own system, making it difficult for classified information to spill from one system to another because the connections between systems can be highly controlled.

With MLS, these systems can be consolidated onto a single system, with each compartment independent of the other, so that transfer of data can occur between compartments within that system only according to strict MLS rules. This takes advantage of the cost savings of not having to manage multiple systems, but only one or a few.

Commercial customers also might find some features of MLS useful, such as to separate sensitive client information from general information or from another user. New government regulations and corporate mergers are examples where security of information based on the information itself is important in the commercial world.

Characteristics of a multilevel-secure system include:

- The system controls access to resources.
- The system does not allow a storage object to be reused until it is purged of residual data.
- The system enforces accountability by requiring each user to be identified, and creating audit records that associate security-relevant events with the users who cause them.
- The system labels all hard copy with security information.
- The system optionally hides the names of data sets, files, and directories from users who do not have access to those data objects.
- The system does not allow a user to declassify data by “writing down” (that is, write data to a lower classification than the one at which it was read) except with explicit authorization to do so.

A multilevel security system is a security environment that allows the protection of data, based on both traditional *Discretionary Access Controls* (DACs), and on controls that check the sensitivity of the data itself through *Mandatory Access Controls* (MACs).

Users access data based on a comparison of the classification of the user and on the classification of the data as well as on the standard DAC checking. This additional security check verifies that users can access only data and resources that their classification allows them to.

MLS is not new to the z/OS security server (RACF). The base features of MLS have been implemented in RACF since 1990. However, at that time MLS was not

addressing specific resources such as the TCP/IP resources or the UNIX System Services resources. This is now addressed in enhancements to RACF MLS.

RACF has several options that can be turned on and off to manipulate different aspects of a multilevel security environment. Because it is possible to have some features of multilevel security on at one time (creating a partial multilevel security environment), commercial customers might find this type of environment useful to meet these needs compared to running a full-fledged multilevel security environment.

MLS is supported by DB2 for z/OS, where it provides row-level security labeling. It is also supported by UNIX System Services (for files and also for Inter-Process Communication) and also by TCP/IP communications.

4.1.5 Auditing and logging

One of the most important features of a centralized authentication and access control mechanism such as RACF is the ability, from a single focal point, to record and analyze security information. This information, called *audit data*, is essential for ensuring that the customer's installation security policy is being followed.

Proper monitoring of the installation requires the analysis of two types of audit information: static information, such as the user registry and access information, and dynamic information, which is a record of security-relevant information such as system or resource access.

The RACF option of the z/OS Security Server provides multiple ways to specify which security-relevant events are recorded in the audit stream and how that information is reduced and analyzed.

In the case of resources, the RACF security administrator or a resource owner (as designated by RACF) may request the auditing of events related to a specific RACF-controlled resource (or set of resources). Similarly, the RACF auditor may request auditing of events related to resources, but independent of the security administrator and the resource owner (that is, entirely at the discretion of the auditor and without the owner necessarily being aware of the audit). Auditors may request the auditing of entire RACF classes of resources or individual resources depending on the need.

In auditing users, the RACF security administrator or the RACF auditor can cause the activities of a specific user to be logged. Such users need not be aware that their activities are being monitored.

Finally, RACF can enforce a separation of duties with respect to security administration and auditing by allowing the administrator to change access control rules and user definitions but not to change certain global auditing controls. Conversely, the auditor can change auditing controls but cannot make administrative changes.

In other words, the administrator cannot prevent the auditor from auditing the administrator, while the auditor cannot authorize himself or herself to resources. The result, when this capability is properly exploited, is that collusion between the two is required to cheat the system.

RACF provides several utilities to assist security administrators and auditors in the analysis of security information.

Each of these utilities provides the auditor with data analysis and reduction capabilities to ensure that users are adhering to the installation security policy.

4.1.6 Networking and communications security

Networking and communications security on System z servers is provided by the Communications Server component of z/OS.

The Communications Server, along with other elements of z/OS, provides the following security functions to address TCP/IP security concerns:

- *Protecting data in the network.* The Communications Server protects data in the network using secure protocols based on cryptography, such as IP Security (IPSec), SSL, and SNA session-level encryption.
- *Protecting system resources and data from unauthorized access.* Communications Server applications and the TCP/IP protocol stack protect data and resources on the system using standard RACF services.
- *Protecting the system from the network*. The Communications Server is responsible for protecting the system against denial-of-service attacks from the network. The Communications Server has built-in defenses and also provides several optional services that an installation can deploy to protect against these attacks.

Protecting data in the network

As TCP/IP applications are rolled out on System z processors within the enterprise, security in the IP network may be required to protect data traffic, depending on the sensitivity of the data and the level of trust that the installation has in the IP network.

If the System z application is not enabled to use SSL or Kerberos, IP Security (IPSec) can be used. Furthermore, as enterprises seek to engage in business-to-business or same-business communications using the Internet as a portion of the data path, IPSec can be used to protect not only the application data but also the IP header information. IPSec can be used to build a Virtual Private Network (VPN) to support these e-business configurations.

IPSec, an IETF-defined standard, provides data privacy, data origin authentication, and data integrity services. Because IPSec is implemented at the network layer rather than in the application (as is the case for SSL), it can protect all applications transparently without requiring applications to be changed.

IPSec protects data traffic using security associations, which provide cryptographic security services for the data traffic they carry. Security associations and associated cryptographic keys can be defined manually, or they can be created dynamically and securely using the Internet Key Exchange (IKE) protocol.

Protecting resources and data from unauthorized access

Communications Server applications use RACF for identification, authentication, and access control decisions. Some applications allow access by an end user who is not identified and authenticated. These applications must be configured specifically to permit anonymous access. When such access is allowed, resources that can be accessed are limited by server configuration, or by using a specific RACF user ID (with limited access privilege) assigned to the anonymous user.

By using an RACF user ID to represent the anonymous user, the application can access only the resources that are permitted according to the installation-defined RACF policy for anonymous users.

In addition to the TCP/IP application support of RACF, the Communications Server TCP/IP protocol stack uses RACF to control local z/OS users' access to TCP/IP resources, such as a specified TCP/IP system, TCP or UDP (User Datagram Protocol) port, or IP network resource (or group of IP network resources).

Other protections are available, such as syslog isolation, which provides a method for segregating system-level and user-level syslog records based on user ID and job name. It also records the real user ID and job name in the syslog record to prevent undetected *spoofing*.

Protecting the system from the network

The z/OS Communications Server has built-in defenses to ensure high availability of the system against denial-of-service attacks from the network.

The installation can request further protection by configuring for *IP packet filtering*. IP packet filtering can permit or discard inbound or outbound packets by matching configured selectors such as IP address, port, and protocol with information in the IP packet.

A traffic regulation management daemon (TRMD) can be configured to control the number of connections that are allowed on a TCP port. The number of connections can affect system resource consumption, such as memory and number of address spaces. TRMD protects the system against a spike in the number of connection requests.

Intrusion detection systems (IDSs) are used to detect potential intrusions and attacks on the network or on a host in the network. There are many types of intrusion detection systems, each with its own advantage. In general, an installation should incorporate multiple IDS types from multiple vendors in order to broaden intrusion-detection coverage.

In z/OS, the Communications Server provides intrusion detection services that are integrated with the TCP/IP stack.

This approach has a number of strengths when compared to intrusion detection systems deployed in the network. These strengths are based on the IDS's exploitation of its location, which is the communications endpoint. The IDS can examine data that is encrypted end-to-end by using IPSec after decryption.

The local IDS also has access to information that is unavailable to a network IDS, such as memory and CPU usage, connection state information, internal data queue lengths, and packet-discard rates and reasons. Using this information, the integrated IDS can detect some attacks that are not detectable using a network-based approach.

The integrated IDS is under policy control to identify, alert, and document suspicious activity. It detects the following types of events:

- Scans (TCP and UDP port scans, ICMP scans)
- Attacks against the TCP/IP stack
- Flooding (both TCP and UDP)

The policy repository can be either LDAP or a flat file.

4.2 z/OS Integrated Security Services

Integrated Security Services are another set of security functions that are part of z/OS, in addition to the security server described in the previous section. The most important of these services are:

- Integrated IPSec/VPN support
- LDAP directory server
- Network authentication service
- Enterprise Identity Mapping

4.2.1 Integrated IPSec/VPN support

In z/OS 1.8 (which is the current release as we write this book), a new component of the Communications Server called the *Integrated IPSec/VPN* provides the necessary functions to support:

- IP filtering
- IPSec
- Internet Key Exchange (IKE)
- NAT-Traversal support (NAT-T), which is a new function that enables some IPSec dynamic tunnels to traverse a NAT (network address translation) device.

It is also worth mentioning that, in z/OS 1.8, the Communications Server supports the Advanced Encryption Standard (AES) for Integrated IPSec/VPN.

AES has been designated by the National Institute of Standards (NIST) as the replacement for DES as the standard encryption algorithm.

4.2.2 LDAP directory server

The lightweight directory access protocol (LDAP) server is a general-purpose distributed directory server implemented in z/OS that can contain many different types of information, such as distributed application descriptions, configuration information, user and group definitions, and so forth.

The z/OS implementation of LDAP is designed to be complementary to RACF and to interoperate with it in support of the integration of the centralized computing model, traditionally supported by RACF, into the emerging distributed computing models, such as those defined by the Common Object Request Broker Architecture (CORBA) and the Enterprise JavaBeans (EJB) environment provided by WebSphere.

The LDAP offering on z/OS includes both an LDAP V3 server and client. The server supports several backends, actually connectors, to the data repositories that the z/OS LDAP server externalizes as though they were LDAP directories. The purpose of this externalization is to allow remote data manipulation from LDAP clients, assuming that the remote user can properly authenticate to the LDAP server. The LDAP server on z/OS is a z/OS UNIX application.

The LDAP server's role in the security environment is one of an easily accessible data store for security information ranging from X.509 v3 digital certificates to access control policy information. The LDAP protocol provides an industry-standard access mechanism to these data.

On z/OS, the LDAP server provides the ability to extend the native security services provided by RACF with distributed security capabilities provided by cross-platform applications and services.

The current LDAP server is a base element of the z/OS Integrated Security Services. Its functions have been "stabilized" since z/OS V1R6.

A new LDAP server will be soon available on z/OS V1R8: the IBM Tivoli® Directory Server for z/OS. It is a redesigned LDAP server, with a simplified environment: for instance, it does not require DB2 anymore as a backend: it uses a file-based repository in the z/OS UNIX File System.

4.2.3 Network authentication service (Kerberos)

Network Authentication Service is a z/OS implementation of a Kerberos Key Distribution Center (KDC). It was made available as a z/OS standard feature several years ago.

There are three main components in the implementation:

- A z/OS UNIX application, that acts as a an Authentication Service (AS) server
- RACF, or an equivalent product, acting as a Kerberos registry
- A Kerberos-enabled application that accepts Kerberos tickets as authentication credentials

The following products are enabled for Kerberos:

- DB2 V7 and above (authentication)
- WebSphere Application Server V4 and above (authentication)
- FTP client and server (authentication, optional encryption)
- Telnet server (authentication, optional encryption)
- LDAP client and server (authentication)
- rshd server (authentication, optional encryption)

4.2.4 Enterprise Identity Mapping (EIM)

Today's network environments are made up of a complex group of systems and applications, resulting in the need to manage multiple user registries. These user registries are intended to be used by applications to achieve user identification and authentication. The authenticated user ID is eventually used for access control as performed by the application itself, or the middleware it runs on, or by the local operating system.

Typically the multiple user registries have a design and implementation specific to each type of platform, or even application, because some of them might have been implemented at the application level. They have grown over time along with the user population and the introduction of new systems and applications, yielding large administrative problems that affect users, administrators, and application developers when it comes to keeping track of a single entity and its many representations in the installation.

Also note that systems-specific local user IDs and registries, with their underlying mechanisms, are not going to unify in a common format across all vendors, at least for the next few years. What seems the most natural way to proceed is for a user to authenticate to an installation using a network identity.

Enterprise Identity Mapping (EIM) is an IBM technology that enables administrators and application developers to address the problem of multiple user identities and registries more easily and inexpensively than previously possible.

EIM accomplishes this by providing a central place to store mappings between user IDs that are defined in different registries in an installation. These mappings indicate:

- A relationship between user IDs (that is, user IDs that belong to the same person or entity within the enterprise)
- A transformation of one or more user IDs to an application-specific user ID

In the EIM terminology, the unique name given at an enterprise level for a user or an entity is the EIM Identifier. EIM actually defines associations between an EIM identifier and user IDs in registries that are part of OS platforms, applications, and middleware.

Applications, typically servers, can then use an EIM API to find a mapping that transforms the installation level user ID initially used for authentication to a local user ID, which can in turn be used to access local resources.

It is important to note that, as the name implies, EIM provides identity mapping only. EIM does not take care of the user authentication itself, which obviously must be performed before identity mapping. It is left to the application

deployment strategy to decide on how user authentication should be performed and how interapplication trust can be implemented. Current Kerberos authentication appears to nicely fit authentication needs in an EIM environment.

Note: EIM is often mistaken for a user ID management solution similar to products such as Tivoli Identity Manager. An identity management solution is able to work with all aspects of a user ID, passwords, and user authorities. EIM contains only the names of user IDs and none of the other attributes. It has insufficient information for authentication or authorization.

The benefits of EIM are:

- The application server does not need to invent yet another user registry; it can use existing protocols for authenticating users and existing resource access managers to control the use of resources.
- The application server does not need to store these mappings in side files.
- The mappings can be used by more than one application.
- The mappings can span different platforms in the enterprise.
- Multi-tier applications can be written that will not ask the user for a platform-specific identity. They can get this identity without going into an identification and authentication process that might expose passwords.

4.3 z/OS cryptographic services

The following sections provide a description of the cryptographic services.

4.3.1 ICSF

The Integrated Cryptographic Service Facility (ICSF) is the z/OS component that drives the System z hardware cryptographic coprocessors and provides the IBM Common Cryptography Architecture (CCA) API for the applications and middleware to invoke the cryptographic services. ICSF has been integrated in the base z/OS for several years.

ICSF supports two main types of cryptographic processes:

- Symmetric, or secret key, algorithms, in which the same key value is used in both the encryption and decryption processes.
- Asymmetric, or public key, algorithms, in which the key that is used in the decryption process is different from the one used in the encryption process.

4.3.2 OCSF

The Open Cryptographic Service Facility (OCSF) is the z/OS implementation of the Common Data Security Architecture (CDSA) API for applications running in the UNIX Services environment.

Practically speaking, this means that the OCSF services are intended to be used by z/OS UNIX System Services–based application servers or daemons. It has been part of the base z/OS for several years.

4.3.3 System SSL

Secure Sockets Layer (SSL) is a client-server communication protocol developed by Netscape for securing communications that use TCP/IP sockets. It uses both asymmetric (public key) and symmetric key cryptography:

- For server authentication
- To provide data privacy and integrity
- For optional client authentication via digital certificate

The SSL protocol is executed *at the application level*; therefore an application has to be designed to support SSL in order to benefit from the SSL protection.

System SSL was introduced several years ago in base z/OS as a set of libraries that are called by APIs provided to C/C ++ applications protecting their TCP/IP sockets communications with SSL. Consequently, these applications only have to properly call the System SSL API, as opposed to having been designed with complete SSL support embedded. These applications can act either as an SSL client or an SSL server, and should call the API functions accordingly.

System SSL also comprises the gskkyman utility, a UNIX utility that creates and manages key pairs and certificates.

SSL is known to be a very demanding protocol in terms of computing resources in the session initialization phase (handshake). These resources are needed to support the heavy calculation required by the asymmetric (that is, public key) cryptography that these handshake uses, hence the advantage of having hardware cryptographic facilities to which these calculations can be offloaded.

System SSL transparently invokes the hardware cryptography, if it is available on the system, to perform the RSA computation required by the handshake and for the data encryption if DES or T-DES have been negotiated between the SSL client and server.

System SSL has been updated repeatedly to extend the protocol capabilities as proposed in existing or new RFCs and to take advantage of new hardware cryptographic coprocessor technology.

4.3.4 z/OS PKI services

z/OS public key infrastructure (PKI) services was introduced as a base component of z/OS several years ago. It is an industry-class Registration and Certification Authority software that runs on z/OS, exploiting the capability of RACF or an equivalent product, to protect the Certification Authority RSA private key and, whenever available, using the hardware cryptographic services.

4.4 System z hardware cryptographic functions

IBM has a long history of providing hardware cryptographic solutions, from the development of Data Encryption Standard (DES) in the 1970s to delivering integrated cryptographic hardware in a server to achieve the U.S. Government's highest FIPS 140-2 Level 4 rating for secure cryptographic hardware.

System z cryptographic functions include the full range of cryptographic operations needed for e-business, e-commerce, and financial institution applications. The System z environment also includes standard cryptographic hardware and optional cryptographic features for flexibility and growth capability.

Two types of cryptographic hardware functionalities are available on System z:

- Synchronous functions
- Asynchronous functions

4.4.1 Cryptographic synchronous functions

Each System z Processing Unit (PU), also called engine, has an assist processor on the chip, in support of cryptography, that is called the *CP Assist for Cryptographic Function* (CPACF). It provides a set of symmetric cryptographic functions that enhance the encryption and decryption performance of clear key operations for Secure Sockets Layer (SSL), virtual private networks (VPN), as well as data-storing applications that do not require FIPS 140-2 level 4 security.

System z hardware includes the implementation of algorithms as hardware synchronous operations (that is, holding the PU processing of the instruction flow until the operation has completed). These algorithms are:

- DES, Triple DES (T-DES) data encryption and decryption
- AES data encryption and decryption (128-bit keys)

- Message authentication code (MAC): single key-MAC and double-key MAC
- Hashing algorithms (SHA-1 and SHA-256)
- Pseudo Random Number Generation (PRNG)

It is important to remember that keys, when used by CPACF, exist for a short period in the main memory of the computer system.

The functions mentioned above are directly available to application programs, because they are provided as problem state z/Architecture instructions. Alternatively, they can be called through the ICSF component of z/OS, by an ICSF-aware application.

4.4.2 Cryptographic asynchronous functions

Asynchronous cryptographic functions are provided by optional Peripheral Component Interconnect Extended (PCI-X) cryptographic adapters that provide a high-performance, secure key cryptographic environment.

These adapters provide the following secure key functions:

- Data encryption/decryption algorithms
- DES key generation and distribution
- PIN generation, verification, and translation functions
- Pseudo Random Number Generation (PRNG)
- Public Key Algorithm (PKA) Facility, for application programs using public key algorithms

In fact, the PCI-X adapter can be configured in either of the following ways:

- As a coprocessor
- As an accelerator

Crypto Express2 coprocessor

When configured as a coprocessor, the PCI-X adapter is called a Crypto Express2 coprocessor, and is compliant with the FIPS 140-2 Level 4 rating for secure cryptographic hardware modules. Among other things, this means that unauthorized removal of or tampering with the adapter results in zeroing its content.

The PCIXCC coprocessor has a full set of specialized hardware circuits to provide high-speed and secure cryptographic functions as follows:

- DES and T-DES engine
- High performance RSA modular arithmetic
- Random Number Generation

- Secure storage for:
  - The symmetric and asymmetric master keys
  - The “retained” RSA private keys
  - The roles and profiles defined to the coprocessor

With the higher level of API implemented via the software layers in the coprocessor itself and ICSF, the hardware-assisted functions that are available to the applications are:

- DES and T-DES encryption
- MAC functions that support ANSI standards X9.9 and X9.19
- TDES-based key management using control vectors
- RSA key generation
- RSA-based digital signatures
- RSA-based key management of DES keys
- SET for electronic commerce functions
- PIN processing functions
- Processing Europay-MasterCard-Visa (EMV) smartcard-based transactions

Other functionalities of the Crypto Express2 coprocessor relate to the security of the public/private key encryption processing. The most important are:

- Remote loading of initial ATM keys. This provides the ability to remotely load the initial ATM keys. Remote Key Loading refers to the process of loading DES keys to ATM from a central administrative site without the need for personnel to visit each machine to manually load the DES keys. The process uses ICSF callable services along with the Crypto Express2 feature to perform the remote load.
- Key exchange with non-CCA systems. Allows for changing of the operational keys between the remote site and the non-CCA system such as the ATM.
- User-defined extensions (UDX) support. UDX enables the user to add customized operations to a cryptographic processor. User-defined extensions to the CCA support customized operations that execute within the Crypto Express2 feature. UDX is supported through an IBM, or approved third-party, service offering.

The security-relevant portion of the cryptographic functions is performed inside the secure physical boundary of a tamper-resistant card. Master keys and other security-relevant information is also maintained inside this secure boundary.

A Crypto Express2 coprocessor operates with the ICSF and the z/OS Security Server (RACF or equivalent software products), in a z/OS operating environment, to provide data privacy, data integrity, cryptographic key generation and installation, electronic cryptographic key distribution, and Personal Identification Number (PIN) processing.

Crypto Express2 accelerator

When configured as an accelerator, the PCI-X adapter is called a Crypto Express2 accelerator. It is actually a coprocessor that is reconfigured by the user to use only a subset of the coprocessor functions, but at a higher speed.

Reconfiguration is done at the PCI-X cryptographic adapter level, that is, a Crypto Express2 feature can host a coprocessor and an accelerator, two coprocessors, or two accelerators.

However, the accelerator operates with clear keys only, so it is not compliant with the FIPS 140-2 certification. Also, User Defined Extensions are not available on the accelerator.

The main functions provided by the accelerator are:

- High-speed SSL handshake
- High-speed asymmetric (clear key) encryption
- Clear key RSA operations

The functions that remain available to the PCI-X adapter when configured as an accelerator are used for the acceleration of modular arithmetic operations (that is, the RSA cryptographic operations used with the SSL/TLS protocol):

- PKA Decrypt (CSNDPKD), with PKCS-1.2 formatting
- PKA Encrypt (CSNDPKE), with Zero-Pad formatting
- Digital Signature Verify

The maximum number of SSL transactions per second that can be supported on a z9 EC by any combination of CPACF and accelerators is limited by the number of cycles that are available to perform the software portion of the SSL/TLS transactions.

When both PCI-X cryptographic adapters are configured as accelerators on a z9 EC, the Crypto Express2 feature is designed to perform up to 6000 SSL handshakes per second.

4.4.3 Reasons for using hardware cryptography

The following sections illustrate some of the potential advantages of using hardware cryptography in the System z environment:

FIPS 140-2 Level 4 Certification/Rating

The PCI-X adapter, whether it is used in System z (where it is called Crypto Express2 coprocessor or accelerator), or in System x™ (where it is called IBM 4764 PCI-X cryptographic coprocessor) is the only product of this category (type) that is rated at such a high level.

This rating also simplifies the transportation, exploitation, and destruction procedures, meaning by that when an installation receives a 4764 from IBM, either in a System x or inside a System z, it is assured of the integrity of this hardware. (If anybody had tried to tamper with the hardware after coming out of the assembly line, it would have become unusable.)

Long-term guarantee for customer investments

IBM defined the Common Cryptographic Architecture CCA in 1989. Since then, it has evolved and been supported by each new mainframe series, providing upward compatibility.

Note: Briefly, CCA is a set of common cryptographic functions across multiple platforms that enable you to encipher data on one system, send it to another system, and decipher it on the destination system. CCA provides:

- Interoperability between applications on different systems/platforms.
- Application portability at the source-code level between systems or platforms

Good match between CCA and requirements

CCA supports the crypto services that are defined by international users (banks) for payment and debit operations. The implementation of those services is particularly secure because of:

- The intrinsic security of the coprocessor
- The specifics of the CCA architecture (for example, unmodifiable characterization of the keys).

This makes the System z platform a popular choice for real-time banking and finance applications that require high security, performance, and availability.

CCA scaling and extensibility

Faced with particular requirements, software editors or customers can add functionalities to CCA by adding UDX services that correspond to national standards or particular needs. This is true, in particular, in the domain of producing or personalizing smart cards, so that they can be put in service with all the required security and safety.

Interoperability

It is important to note that the 4764 technology and the CCA API are available on all IBM platforms: System x, System i™, System p™, and System z. Thus, installations can deploy or consolidate on System z banking applications that come from other platforms.

Installations can also deploy the various components of a banking (or money transfer) application on the platforms of their choice (for example, personalization of banking cards on System x, key management on System p, and real-time money transfer applications on System z), all while benefiting from a consistent security level across the various platforms involved (this is very important), as well as from interoperable key exchange and secure data transfer protocols.

4.5 IBM Encryption Facility for z/OS

Network traffic flowing over the Internet or an intranet can be protected through security protocols such as Secure Sockets Layer (SSL) and Transport Layer Security (TLS), but encrypting data that resides on physical media such as tape or disk (sometimes referred to as “data at rest”) is growing in importance for any business that deals with sensitive information.

For more than 15 years mainframes have enabled encryption of data traveling across a network. However, to answer customer concerns about data privacy, regulatory compliance, and loss or tampering of private data on removable media, IBM announced in September 2005 a set of functionalities to encrypt and decrypt data on removable media that leverage on z/OS strengths such as hardware-assisted cryptography (as described in the previous section) and centralized key management in ICSF.

These functions are grouped into the IBM Encryption Facility for z/OS., which covers three products:

- IBM Encryption Facility for z/OS, Services feature
- Encryption Facility Client
- DFSMSdss Encryption Feature

Encryption Facility Client is downloadable from the Web, free of charge. The other two products are chargeable.

Encryption Facility for z/OS uses ICSF to perform encryption and decryption of data and to manage cryptographic keys. Those keys can be of two types:

- T-DES triple-length keys (clear or secure keys)
- 128-bit AES keys (clear keys only)

Both features can use the state-of-the-art encryption and centralized key management capabilities provided by functions of z/OS and features of System z and eServer zSeries servers to help secure data that has been stored to tape and other removable media.

Statement of direction

The facility just presented is called “inboard encryption” because the encryption is performed on the server side. “Outboard encryption” provides similar functions within the storage environments such that the capability does not require the use of host server resources.

The statement includes the intent to support outboard encryption and to leverage the centralized key management provided by z/OS ICSF.

The first implementation of outboard encryption is planned for the IBM System Storage™ TS1120 tape drive in the second half of 2006.

Plans include a new software program for management of encryption keys for tape drives across the enterprise. The new software might use standard key repositories, including ICSF on z/OS.

Figure 4-1 illustrates both options, the current and the planned capabilities:

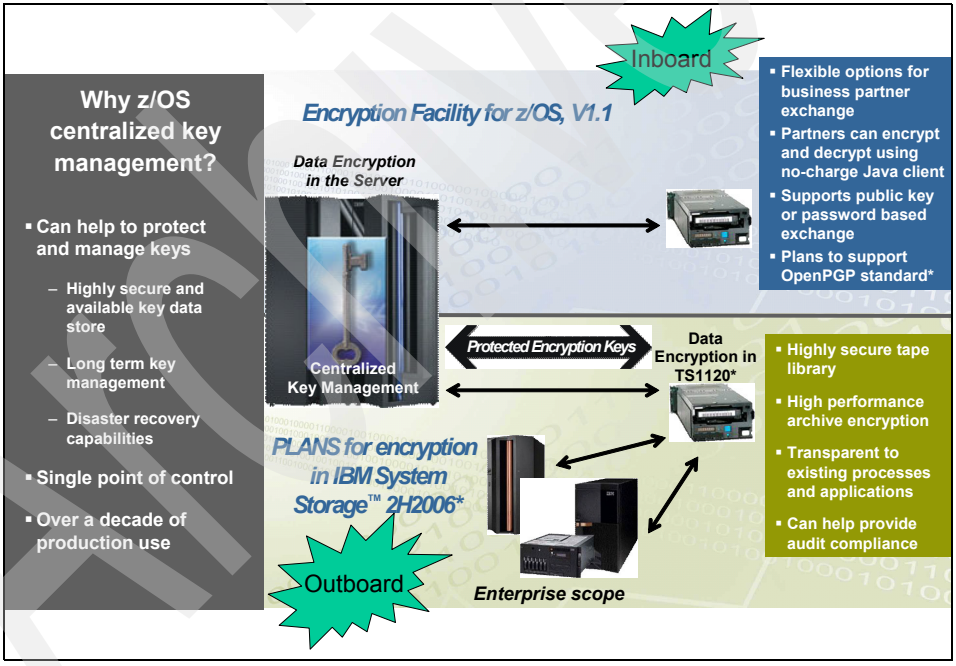


Figure 4-1 Removable media encryption on System z platform

4.5.1 IBM Encryption Facility for z/OS, Services feature

This optional priced product, also called IBM Encryption Facility for z/OS Encryption Services, or Encryption Services, supports encryption and decryption

of sequential files, members of partitioned data sets (PDS), partitioned data sets extended (PDSE), and files from z/OS UNIX System Services. Using System z hardware compression features, the Encryption Services feature can optionally compress the data before encrypting it. This feature can use ICSF and the hardware cryptographic capabilities of the System z platform.

4.5.2 Encryption Facility Client

This downloadable product, sometimes also called Encryption Facility for z/OS Client, is a Java package that provides the capability to encrypt and decrypt z/OS format data on a non-z/OS platform, or on z/OS systems that do not use Encryption Services.

In other words, it enables exchanging encrypted data between z/OS and non-z/OS platforms if the encrypted data is created through Encryption Services or Encryption Facility for z/OS Client.

However, Encryption Facility for z/OS Client cannot uncompress data that was compressed by Encryption Services. (Software decompression is very slow, so this would not make sense.) This why compression is optional when using Encryption Services.

Figure 4-2 illustrates the process.

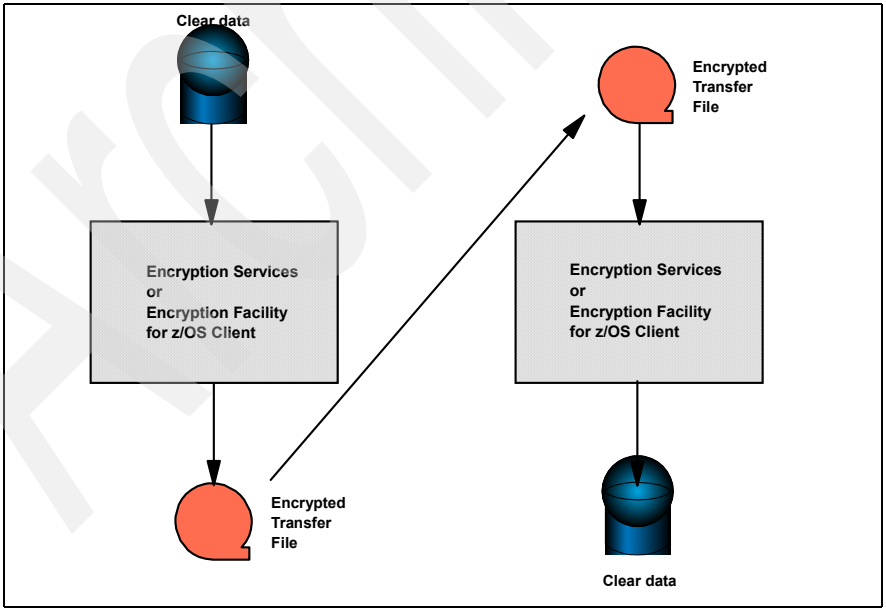


Figure 4-2 Encryption facility process

4.5.3 DFSMSdss Encryption feature

This optional priced product enables use of z/OS DFSMSdss to encrypt data transferred through the DUMP command to tape or DASD, and, similarly, to decrypt data transferred through the RESTORE command.

Figure 4-3 illustrates the process.

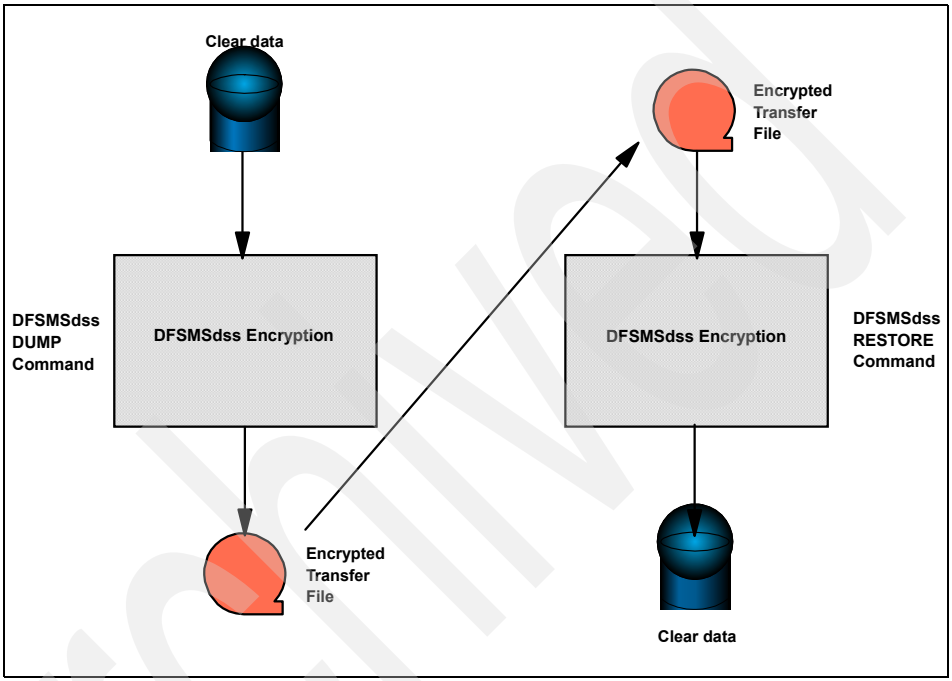


Figure 4-3 DFSMSdss encryption process

4.6 IBM System Storage TS1120

Although it is not strictly related to the System z platform, the recently announced IBM System Storage TS1120 tape drive, with built-in hardware encryption, is supported on the System z platform (as well as on all other IBM platforms and on some non-IBM platforms) and provides an interesting complement to the IBM Encryption Facility for z/OS.

On z/OS the encryption is “system-managed” and requires the Encryption Key Manager (EKM), which is a Java application running on the System z host. EKM, which is supported by DFSMS, generates and manages the 256-bit AES keys.

The TS1120 tape drive uses both symmetric and asymmetric encryption algorithms:

- Symmetric encryption for high-speed encryption of user or host data
- Asymmetric encryption (which is slower) for protecting the symmetric key that is used to encrypt the data (called *key wrapping*)

When unencrypted data (clear text) is sent to the TS1120 tape drive for encryption, it is converted to ciphertext through AES encryption, a symmetric (or secret) key type of encryption requiring a symmetric Data Key (DK), and is then written to tape.

The 256-bit AES Data Key is also encrypted, or wrapped, using the public key from an asymmetric Key Encrypting Key (KEK) pair to create an Externally Encrypted Data Key (EEDK). This EEDK is written to the cartridge memory and to three additional places on the IBM 3592 Tape Cartridge. The tape cartridge now has both the encrypted data and the means to decrypt it for anyone who holds the private KEK.

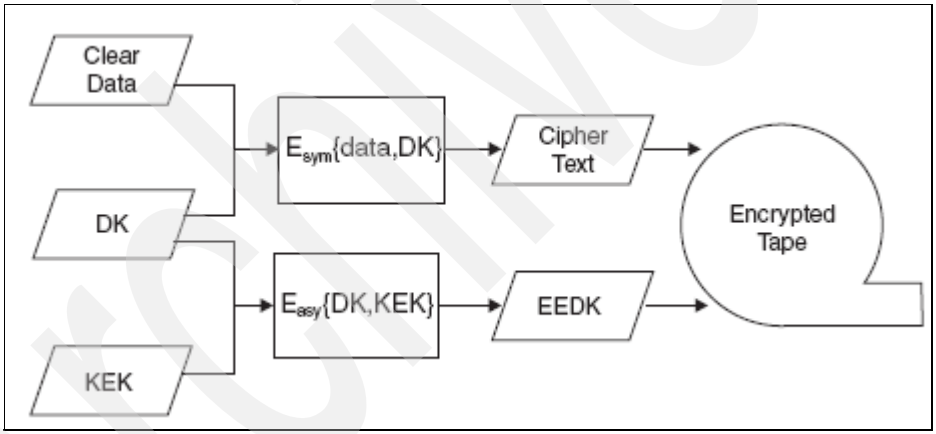


Figure 4-4 EKM process

When the EKM runs on z/OS, it works in conjunction with ICSF to manage the encryption keys.

4.7 Security certifications

To address the need for security certification of IT systems worldwide, and, as a result of the progressive convergence between the NSA “Orange Book” in the USA and the ITSEC standards in Europe, the Common Criteria for Information

Technology Security Evaluation, also known as ISO standard 15408 (published in 1999), was developed by the following national security organizations:

- CSE (Canada)
- SCSSI (France)
- BSI (Germany)
- NLNCSA (Netherlands)
- CESG (UK)
- NIST (USA)
- NSA (USA)

These Common Criteria define seven levels of evaluation, called Evaluation Assurance Levels (EALs), as follows:

EAL1	Functionally tested
EAL2	Structurally tested
EAL3	Methodically tested and checked
EAL4	Methodically designed, tested and reviewed
EAL5	Semi-formally designed and tested
EAL6	Semi-formally verified design and tested
EAL7	Formally verified design and tested

Some additional information helps explain the security certification environment, such as:

- The EAL rating is not meaningful in itself; it has to be related to the Protection Profile used in establishing the rating.
- A Protection Profile is an implementation-independent set of security requirements for a category of Targets of Evaluation that meet specific consumer needs. Examples of Protection Profiles are:
  - Controlled Access Protection Profile (CAPP)
  - Labeled Security Protection Profile (LSPP)
- A Target of Evaluation (TOE) is an IT product or system and its associated administrator and user guidance documentation that is the subject of an evaluation.
- Federal Information Processing Standards (FIPS) are a set of various information processing standards for use within U.S. Government agencies.
- Federal Information Processing Standard 140-1 (FIPS 140-1) and its successor FIPS 140-2 are U.S. Government standards that provide a benchmark for implementing cryptographic software. They specify best practices for implementing crypto algorithms, handling key material and data buffers, and working with the operating system.

4.7.1 Hardware certifications

This section lists the security certifications for the Crypto Express2 coprocessor and for PR/SM.

The Crypto Express2 coprocessor features now hold the industry's top hardware rating: FIPS 140-2 Level 4.

This certification means that the Crypto Express2 coprocessor security module satisfies the requirements for a cryptographic module utilized within a security system protecting Sensitive Information (United States) or Protected Information (Canada) within computer and telecommunications systems.

To achieve FIPS 140-2 Level 4 certification, an independent laboratory is permitted to attempt virtually any physical attack on the product and must verify the security of the internal software using a mechanical verification of a mathematical model.

PR/SM LPAR for the IBM System z9-EC and z9-BC was evaluated under the Common Criteria at EAL5. The certificate was published on September 4th, 2006.

4.7.2 Software certifications

This section lists the security certifications for the major operating systems used on the System z platform.

z/OS

- z/OS V1.6 was evaluated under the Common Criteria at Evaluated Assurance Level 3+, using both the CAPP and the LSPP. The certification report was published on March 9, 2005.
- z/OS Version 1.7 was evaluated under the Common Criteria, using the CAPP and the LSPP, at Evaluated Assurance Level 4+. The certification report was published on March 2nd, 2006.

z/VM

z/VM V5.1 was evaluated under the Common Criteria at Evaluated Assurance Level 3+, using both the CAPP and the LSPP. The certification report was published on October 26, 2005.

Linux

- IBM sponsored the Common Criteria evaluation of SUSE Linux Enterprise Server Version 9, with certification-sles-ibm-eal4 package. The evaluation

was performed using the CAPP, and achieved an EAL4+ level. The certification report was published on March 9, 2005.

- Red Hat Enterprise Linux 3, Update 2 on IBM Systems servers was evaluated under the Common Criteria, using the CAPP, achieving an EAL3+ level. The evaluation results, announced August 03, 2004, are for Red Hat Enterprise Linux WS on System x platform and Red Hat Enterprise Linux AS on System x, System i, System p, and System z servers as well as Opteron-based systems.
- IBM is sponsoring the in-progress evaluation of Red Hat Enterprise Linux 4, Update 2, under the Common Criteria with a conformance claim of EAL4+.

5

Resiliency

Resiliency is considered one of the most important value propositions of the IT platform, and business strategists usually consider System z as the most resilient platform. This chapter starts by discussing the cost of downtime and the need for resiliency, and then continues by discussing the core strengths that make the System z platform resilient, including:

- Core hardware strengths that enable a System z machine to deliver world-class availability
- Features that help a business eliminate planned outages for both hardware and software changes
- z/OS design features, Parallel Sysplex, and GDPS, and how those solutions enhance the resiliency of the System z platform

5.1 Built for business

The cost of downtime is high, forcing businesses to strive to be resilient to outages. A business must recover quickly from all interruptions, whether planned or unplanned. Almost nothing is worse than being hit by a system outage, even if the duration of the outage is only minutes long. Consider how an unexpected outage, caused perhaps by a natural disaster or a minor software glitch, can put customer relationships, productivity, and possibly billions of dollars at stake if vital data and applications go unprotected.

Business resilience, which can be considered the ability of an enterprise to continue to function effectively in the face of natural and man-made problems and disasters affecting its IT, has been one of three most important value propositions of IT itself, along with competitive advantage and cost savings. Business strategists usually consider the System z environment to be the most resilient platform, whose resiliency can be counted on but must be integrated with other systems to achieve a business's resiliency objectives. The System z platform, with its strengths in availability and reliability, is an ideal platform from which to build a resilient infrastructure and from which to test new resiliency technologies and strategies.

Some computer platforms might have been designed originally for academic or other purposes, but the System z environment was created as a solution for business. Over the years, resiliency technologies have evolved and become deeply embedded into System z design. Today, the z platform continues to innovate and enhance its resilient technologies. Resiliency is imperative to a business' bottom line, and System z is the premier platform in ensuring business systems stay up through disaster recovery, repair and upgrade, and also software and application changes.

5.1.1 A solution for continuous business operation

Downtime is costly. The cost of downtime is so great that many of today's enterprises can no longer afford planned or unplanned outages. The statistics are staggering as businesses can potentially face losses from tens of thousands of dollars to multiple millions of dollars per hour of downtime. Even beyond the financial aspects, downtime can also affect key areas of customer loyalty, market competitiveness, and regulatory compliance. For example, if a Web site is not responsive, Internet-savvy customers will go elsewhere. Continuous operation is a business imperative, and it requires having state-of-the-art resilient technology.

Table 5-1 Downtime costs per hour for various industries[1]

Industry	Downtime cost (per hour)
Brokerage Retail	$6.5 million
Credit Card Sales Authorization	$2.6 million
Airline Reservation Centers	$90,000
Package Shipping Services	$28,250
Manufacturing Industry	$26,761
Banking Industry	$17,093
Transportation Industry	$9,435

Some businesses must shut down their systems to make scheduled updates, perform maintenance, or upgrade their servers. Built-in features on the System z platform, however, enable businesses to avoid those scheduled outages. In addition to being an excellent platform solution for eliminating planned outages, System z is also an excellent platform for avoiding unscheduled outages. With technologies such as Parallel Sysplex and GDPS, System z can position a data center for world-class disaster recovery.

5.1.2 World-class availability and reliability

Every minute, the pace of the business world increases and the volume of data grows. Businesses need to keep operations running 24x7 and overcome issues such as higher network traffic, unpredictable workload spikes, and unwieldy databases. System z hardware, firmware, middleware, and operating systems are designed and tightly integrated to provide an application environment with world-class levels of high availability. System z provides many of the best-of-breed technologies to ensure availability and reliability.

System availability can be measured in different contexts. One common measurement is hardware availability against unplanned outages. A more stringent measurement of availability examines application availability at the user level, including planned outages. Measuring availability from the user perspective requires consideration of the hardware, OS, middleware, and applications. System z holds itself to the strictest measurements and can achieve world-class application availability at the user level.

[1] Eagle Rock Alliance, LTD. 2003.

5.2 Core hardware strengths

System z machines, built for recovery and continuous operation, are highly available servers. The result of the focus on resiliency is a design point called Mean Time Between Failure (MTBF). The System z product line is designed to offer layer upon layer of fault tolerance and error-checking features. If a failure occurs, the redundancy that is built into the platform shifts the work from failing components to ones that work to prevent the application and user service from being interrupted. In addition, failed components may be removed and replaced while applications are active and continue to run, eliminating downtime.

Errors have many possible causes. There are transient errors, those due to environmental conditions such as noise or cosmic particles, and more permanent errors such as hardware failures. System z products are designed to handle these various errors. Error detection and correction is built into the logic to detect errors in memory and correct them. Similar technologies help find problems at the earliest possible moment and enable the System z environment to take the necessary actions to correct them, thereby helping to minimize the impact they might have on applications. The System z strategy also focuses on a recovery design to mask errors and make them transparent to customer operations. Extensive recovery is built into the hardware.

Over the years, more and more features have been added that improve the resiliency of the System z platform. The most recent version, System z9 hardware, has a number of unique new features that highlight the newest innovations in resiliency technology.

5.2.1 Redundancy

Redundancy is perhaps the most prevalent mechanism to ensure resiliency, and is a key theme throughout the System z platform. System z designers have worked through the years to eliminate any single point of failure. The platform has many redundant components. For example, there are two power feeds that enable the system to survive the loss of one. There are two MRUs (Modular Refrigeration Units). The service subsystem, internal battery, oscillator, and all of the processors are redundant. Consider a fully loaded System z9 mainframe that has 54 central processors, eight SAPs, two spare CPs, and 336 PowerPC® engines for the FICON channels, which would total 400 processors. After taking into account redundancy, that same system would have 800 processors.

5.2.2 Eliminate planned outages with concurrent hardware changes

In the past, a business often had to take a system down on a Saturday night to do hardware change and maintenance. Today, the System z platform eliminates

such planned outages with its capability to concurrently change hardware without affecting the application. The concurrent hardware maintenance capability helps System z to provide service to its users at any time, 24x7, without scheduled outages for maintenance of system and data.

In today's business world, backups and system maintenance, such as upgrades or replacements, must be done without interrupting operations. Many System z components can be maintained and upgraded concurrently, including: processor books, memory, cryptographic cards, and coupling links. Even non-redundant components such as channel adapter cards are hot-pluggable, and can be replaced without any application impact.

Capacity on Demand

The resilient technologies of System z make it well suited for various Capacity on Demand solutions, offering companies the flexibility to rapidly increase or decrease computing capability as requirements change. To achieve near-continuous availability, System z Capacity on Demand offerings tap into the additional capacity. The flexibility of upgrading processing power concurrently enables a business computing infrastructure to better manage risk of volatile, high-growth, and high-volume applications.

System z Capacity on Demand offerings include:

- Capacity Upgrade on Demand (CUoD): CUoD enables a customer upgrade capacity for nondisruptive growth. With CUoD, a customer can upgrade general CPs, specialty engines, memory, and I/O. The extra capacity can be brought online without an outage.
- Customer Initiated Upgrade (CIU): CIU provides the customer with the ability to upgrade the number of general CPs, specialty engines, and memory.
- On/Off Capacity on Demand (On/Off CoD): On/Off COD provides the customer with the ability to temporarily upgrade capacity for fluctuating workloads. One potential use for this offering would be to handle peak loads. The customer may temporarily upgrade the number of general CPs and specialty engines.
- Capacity BackUp (CBU): CBU is a temporary capacity upgrade for customers who require a robust disaster-recovery solution. With this solution the customer can maintain a hot backup site with limited capacity. In an emergency the extra CBU capacity can be brought online without an outage, bringing the backup site up to a capacity that is capable of taking over the whole workload from the failed production site.

5.3 Beyond hardware

Resiliency in the System z platform goes far beyond the hardware. The System z resiliency design point focuses on the applications, which results in an integrated environment where hardware, firmware, operating systems, and middleware work together to provide application and data availability.

z/OS, the platform's primary operating system, has been built around a philosophy of reliability that recognizes that there will be software errors. Every system service routine has a recovery routine associated with it. In the case of an abnormal end to a program, the recovery routine gets control to repair the data structures used by the failing component and ensure that subsequent calls to this service will not fail again. Based on lines of code, roughly one-third of the z/OS code is associated with the goal of providing reliability, availability, and scalability.

In addition to the primary z/OS system, the z/VM operating system also has many built-in resiliency features. Those familiar with Linux on System z probably have heard about the user who started, as a proof-of-concept exercise, thousands of instances of Linux in thousands of virtual machines on a predecessor of System z9. Although the number of Linux instances is impressive, what is often overlooked is the reliability demonstrated by VM. When VM ran out of resources, it did not crash.

For even more stringent requirements, System z can provide even higher levels of resiliency with its Parallel Sysplex and GDPS architectures. These architectures extend the platform's resiliency technologies to multiple servers and multiple sites. A properly designed Parallel Sysplex with GDPS should be able to provide 100% application availability.

5.3.1 z/OS

z/OS includes a comprehensive set of capabilities that provides its unique qualities of service in combination with System z hardware. With z/OS, most outage events are completely masked from applications, and in severe cases result in graceful degradation rather than complete failure. z/OS has unmatched availability, reliability, scalability, flexibility, and integrity. Therefore, it has been long trusted for database and transaction workloads, and is also a top choice for Web applications that require the highest quality of service.

The z/OS design philosophy dictates a comprehensive approach of error isolation, identification, and recovery. All z/OS components and subsystems provide recovery routines. Furthermore, all z/OS code is written to a set of reliability, availability, and serviceability guidelines. By design, if an application fails, workload can be picked up by a backup of the same application that is running on the same

physical hardware. In addition, z/OS provides a feature that allows for automatic recovery and restart of services and transactions based on predefined policies.

z/OS system software also has a total commitment to system integrity. Using techniques such as storage keys and multiple address spaces, data and system functions are protected from unauthorized access. With a system-wide two-phase commit, client application data is recoverable even when hardware and software failures occur. Some features are also designed to directly support key middleware such as CICS, which enable higher levels of availability for transaction execution.

5.3.2 Planned outages elimination

In addition to the many other methods that are in place to avoid unplanned downtime, System z provides methods to address planned firmware driver updates and reconfigurations.

Customers who run mission-critical applications, usually those without a sysplex environment, have found it difficult to find time to add new functions. On System z, a feature referred to as Enhanced Driver Maintenance (or Concurrent Driver Upgrade) enables a customer to upgrade firmware to the next driver level without having an impact on workload. Enhanced Driver Maintenance allows upgrades of the Licensed Internal Code for CPs, IFLs, ICFs, zAAPs, memory, and I/O adapters that are transparent to the application.

It is also possible to make I/O configuration changes without having a scheduled outage. This reconfiguration capability is called *Dynamic I/O Reconfiguration*, and allows adding, deleting, or modifying the definitions of channel paths, control units, and I/O devices to the software and hardware configurations.

5.3.3 Parallel Sysplex

Parallel Sysplex builds on the strengths of System z to provide even greater availability and even more flexibility to address planned and unplanned outages. Parallel Sysplex is not a single product, but a collection of cooperating z/OS systems. Parallel Sysplex's unique clustering approach makes the multi-server scale-out topology look and behave like a massively parallel SMP (Symmetric Multiprocessing) configuration.

The Parallel Sysplex architecture allows clustered System z servers to provide resource sharing, workload balancing, and data-sharing capabilities for on demand data centers, delivering ultimate flexibility when supporting different application topologies.

You could think of a Parallel Sysplex as a symphony orchestra, with each kind of instrument representing a different product in the sysplex. There are several of

each instrument, just as there would be several images of the same product in a sysplex. All violins, for example, sound basically the same and play the same part. There are enough violinists so that if one is out sick, it will likely not be noticed. And if that violinist quits for good, he can be replaced. Similarly in a sysplex, all systems, or a subset of them, can be made to look alike and do the same work. A sysplex exhibits analogous availability characteristics: A failure or planned removal of one system from a sysplex would not result in the loss of application or data availability, only temporary loss of capacity.

Another advantage of Parallel Sysplex technology is the ability to nondisruptively install or maintain hardware and software. Servers can be removed or added while applications continue to run on the other systems. Furthermore, different systems can be running at different software levels, thereby making software upgrades easier. Software can be upgraded on one system while the rest of the systems remain available. Major software changes can be made using the "rolling IPL" technique to stage the software change throughout the sysplex without ever interrupting the application availability.

With Parallel Sysplex clustering and its ability to support data sharing across servers, IT architects can design and develop applications that have one integrated view of a shared data store, effectively eliminating the need to partition databases. Data sharing with Parallel Sysplex has the unique advantage of allowing nondisruptive database growth with automatic load re-balancing. Without a partitioned database environment, application and/or database growth would likely require lengthy and disruptive database re-partitioning and mean downtime for the application. Parallel Sysplex data sharing capabilities help avoid the availability obstacles encountered with partitioned database architectures.

Parallel Sysplex technology is exploited by System z middleware to provide continuous operation for the applications. Current exploiters from IBM include: CICS Transaction Sever, DB2 for z/OS, WebSphere MQ, WebSphere Application Sever, IMS Transaction Manager, TCP/IP, VTAM®, APPC/MVS, and RACF.

System-Managed Coupling Facility (CF) Structure Duplexing

One self-healing attribute of a Parallel Sysplex is System-Managed CF Structure Duplexing. System-Managed CF Structure Duplexing is designed to provide a hardware-assisted mechanism for duplexing CF structure data, without requiring any involvement with the application or database subsystem. The redundancy of duplexing results in faster recovery because data is already in a second CF when a failure occurs. This feature can provide a robust recovery mechanism for failures such as loss of a single structure or CF, or loss of connectivity to a single CF, through rapid failover to the other structure instance of the duplex pair.

The application and the database manager are not aware that the structure is being duplexed or whether one of the duplex copies has failed or recovered.

5.3.4 GDPS

Picture the average computer room immediately following a basic system failure. Phones are ringing, managers are moving to find out when everything will be recovered, operators are scrambling for procedures, and systems programmers are vying with operators for control of the consoles. Now imagine instead a scenario where the only manual intervention is to confirm whether to proceed with a well-tested recovery procedure. The smooth recovery scenario requires a business continuity solution such as GDPS.

GPDS, the premier IBM solution for high availability, provides business continuity solutions both for system outages and disaster recovery situations. Based on geographical separation and advanced automation techniques, GDPS is a multi-site application availability solution that provides the capability to manage remote copy configuration and storage subsystems, automate Parallel Sysplex operation tasks, and perform failure recovery from a single point of control. It extends the resource sharing, workload balancing, and continuous availability benefits of a Parallel Sysplex environment. It also significantly enhances the capability of an enterprise to recover from disasters and other failures and to manage planned exception conditions.

GDPS is a solution for environments that need to protect data and minimize operational risks. Two key business requirements that should be analyzed when considering GDPS are:

- *Recovery Time Objective (RTO)*, the time a business can afford to wait for IT services to be resumed after a disaster occurs.
- *Recovery Point Objective (RPO)* represents the amount of data that a business is willing to restore in the event of a disaster. For example, for an RPO of six hours, the objective would be to restore systems back to the state they were in six hours ago.

Both RTO and RPO are important decision factors when designing a GDPS solution. Any resource that cannot be replaced within the RTO should be available in multiple locations, and this incorporates not just buildings and hardware but also employees and data. Customers who desire an RPO of zero, which means zero data loss, will require a synchronous remote copy solution, such as that provided by GPDS/PPRC, one of three different GDPS solutions.

The GDPS solutions, which differ based on their mirroring technologies, are:

- GDPS/PPRC

  Based on Peer-to-Peer Remote (PPRC) copy, GDPS/PPRC synchronously mirrors data residing on a set of disk volumes to a secondary site up to 100 km away.

GDPS/PPRC is capable of the following attributes:

- Near-continuous availability
- Near-transparent disaster recovery
- RTO less than an hour
- RPO of zero
- Protects against localized area disasters (limited to 100 km fiber)

- GDPS/XRC

  GDPS/XRC is a combined hardware and z/OS software asynchronous remove copy solution, and is capable of the following attributes:

  - Disaster recovery solution
  - RTO between one to two hours
  - RPO less than one minute
  - Protects against localized and regional disasters
  - Minimal remote copy performance impact

- GDPS/Global Mirror

  GDPS/Global Mirror uses asynchronous PPRC replication technology and is capable of the following attributes:

  - Disaster recovery solution
  - RTO between one and two hours
  - RPO less than one minute
  - Protect against regional disasters
  - Minimal remote copy performance impact
  - Support for z/OS and open data

Exclusive to GDPS in the PPRC environment is HyperSwap™, a function designed to broaden the near-continuous availability attributes of GDPS/PPRC by extending the Parallel Sysplex redundancy to disk subsystems. HyperSwap is designed to swap a large number of devices and do so with minimal impact on application availability.

Additionally, GDPS/PPRC technology can manage a heterogeneous environment of z/OS and Open Systems data. This is important for the common situation when a multi-tier application has dependencies on multiple operating system architectures.

6

Systems management

This chapter discusses the principles of systems management and its evolution from a highly manual process to one where the computer can adapt and change the system based on business policy.

6.1 Systems management overview

The IT environment is more complex than ever and the resources that are required to manage these systems are increasingly hard to find. The increased complexity of hardware and software in modern systems have reached a point where humans cannot reasonably manage them with conventional methods.

Systems management is the enterprise-wide management of computer systems including hardware, operating systems, middleware, applications, and storage subsystems. Systems management consists of the processes, techniques, and tools that are required to manage those systems. System z hardware and software provide a robust set of system-management tools that help IT professionals monitor, manage, and automate their systems.

Systems management, one of the strengths of z/OS, provides a large portion of the value of z/OS. In the mixed workload environment supported by System z and z/OS, system resources are utilized at a high capacity, eliminating wasted resources such as idle CPUs.

The goal of z/OS system management is to make z/OS easier to use, to increase business application availability, and to reduce the skill requirement and cost of managing the system by providing an end-to-end solution.

6.2 The value of systems management on z/OS

z/OS has a long history of systems management and it is one of the most valuable strengths of System z. System z has the ability to perform hardware and software maintenance and installations in a non-disruptive manner with a Parallel Sysplex.

6.2.1 High availability and managed upgrades

With systems configured as a Parallel Sysplex, systems can be removed or added dynamically to the cluster allowing the installation of new hardware or the maintenance of existing hardware. Meanwhile, the remaining systems continue to process work.

IBM also has a software and hardware co-existence policy that enables software and hardware upgrades to be introduced one system at a time. z/OS will co-exist with new hardware with up to three previous releases of z/OS. This enables customers to add new hardware without upgrading the operating system on all machines in the cluster.

IBM middleware also will co-exist with one previous version, giving software architects the ability to decide when to upgrade the middleware based on business policy.

z/OS also provides the ability to perform rolling hardware and software maintenance in a non-disruptive manner. This enables businesses to implement critical business function and react to rapid growth without affecting the availability of the system. z/OS systems in a sysplex cluster can co-exist at different operating levels, up to three release levels apart. This is called a *rolling upgrade*. The new release would be put on the least-critical system first and then rolled to the other systems. This enables the customer to first install a new release on a development system and then roll to a test system, and then eventually roll to a production system. With careful planning it is possible to avoid a full sysplex IPL. By bringing down one system in the sysplex at a time, you will always have access to an MVS image to perform maintenance or emergency repair to the downed system.

6.2.2 Integrated accounting data

A data center has to keep track of resources to determine its expenses so that they can be accounted to the business units. In a distributed environment, a one-to-one ratio of application to server is usually maintained to enable the data center to keep track of these expenses. This model can be expensive because of underutilized system resources, increased data center footprints, and increased power and cooling expenses. To maintain accounting information, applications must be isolated on servers, so at any given time a large portion of the resources of that server might be sitting idle. On z/OS, if one application is idle, the resources can be allocated to another workload. System Management Facility (SMF) provides the capability to keep track of how many resources each workload is consuming. Because multiple workloads can utilize the same resources, z/OS typically runs at a much higher utilization than other IT architectures. The information that SMF maintains can be used for the following tasks:

- Billing users
- Reporting reliability
- Analyzing the configuration
- Scheduling jobs
- Summarizing direct access volume activity
- Profiling system resource use
- Maintaining system security

SMF generates records that can be used to establish availability statistics, maintain configuration change logs, perform data-trend analysis for scheduling jobs based on concurrent activity, job throughput, turnaround time, and idle time, and so forth. SMF provides the building blocks for z/OS product level systems

management. SMF is discussed in 3.3.4, “System Management Facility (SMF)” on page 46.

For more information about SMF, see *z/OS V1R8.0 MVS System Management Facilities,* SA22-7630-13.

6.2.3 Resource management

The Resource Measurement Facility (RMF) is the resource measurement and management standard on z/OS. RMF is designed to simplify the management of single or multiple system workloads. With RMF, a potential bottleneck can be detected earlier and system delays can be avoided or lessened. RMF uses SMF records to perform performance measurement and tuning for resources such as CPU, storage, paging facilities, I/O devices, Coupling Facilities, tape devices, virtual storage usage, XCF buffers, and system ENQs.

RMF provides reporting functions that are needed for managing the performance of a sysplex from a single point. RMF uses goal-oriented reporting to determine whether the system is meeting its performance goals. RMF provides a suite of I/O subsystem reports to manage storage subsystems and devices in the sysplex.

RMF has functionality to load RMF data into spreadsheet applications and to integrate historical performance data in business processes.

RMF integrates with IBM Tivoli System Automation for z/OS and System Display and Search Facility (SDSF) for effective systems automation and work management.

RMF is also discussed in 3.4.1, “Resource Measurement Facility (RMF)” on page 50. For more information about RMF, see *z/OS V1R8.0 Resource Measurement Facility (RMF) User’s Guide,* SC33-7990-11.

6.3 IBM autonomic computing initiative

Computer systems do not become autonomic overnight. This is a process that evolves over time. Best practices are determined through trial and error. For a computer to be able to be autonomic, the logic of these best practices must be embedded into the software logic. Autonomic computing systems evolve over time as system designers become better at predicting and fixing these problems. Businesses slowly incorporate autonomic principles into their IT infrastructure. *Autonomic maturity* is the continuum of the self-managing properties of a system. There are five levels of autonomic maturity: basic, managed, predictive, adaptive, and autonomic.

- *Basic:* IT professionals manage each infrastructure element independently.
- *Managed:* System management technologies are used to collect information onto fewer consoles.
- *Predictive:* Analytical processes are introduced into the system to monitor situations that arise and analyze the situations to provide possible courses of action, but the IT professional decides what action to take.
- *Adaptive:* The IT environment can take actions based on available information and a knowledge of what is happening in the environment.
- *Autonomic:* Business policies and objectives govern the IT infrastructure operation. IT professionals monitor business processes, not IT processes, and alter the objectives of the system based on business policy.

There are examples of predictive and adaptive systems on the market today, but most systems remain at the basic or managed level. z/OS is continually building on the autonomic capabilities of the system with the goal of having a fully autonomic system.

A system at the highest level of autonomic maturity will have the ability to manage itself and dynamically adapt to change in accordance with business polices and objectives. IT personnel set these policies and objectives, which the system acts to fulfill. Today's System z products incorporate a variety of autonomic capabilities based on the four characteristics of self-managing systems: self-configuring, self-healing, self-optimizing, and self-protecting.

- *Self-Configuring:* A self-configuring system has the ability to dynamically configure itself in response to changing technological or business conditions. A self-configuring system will adapt to changing IT environments based on policies set by the IT personnel.
- *Self-Healing:* A self-healing system can detect hardware or firmware faults proactively or as they happen and contain the effects of the faults within defined boundaries. This enables the system to recover with minimal or no impact to the execution of the operating system or user-level workloads. A self-healing system can also recover automatically if an abend or other system error occurs.
- *Self-Optimizing:* A self-optimizing system can detect performance problems by monitoring and correct them through tuning the hardware or software resources.
- *Self-Protecting:* A self-protecting system can detect hostile behavior as it happens and take corrective action to prevent harm to the system. A self-protecting system can protect against internal and external threats to the integrity and privacy of applications and data.

z/OS provides system management that either eliminates the need to take management actions, automates as many management activities as possible, or makes those actions that cannot be automated as intuitive and easy as possible.

6.4 Self-configuring technologies

z/OS is not fully self-configuring yet, but it does have *dynamic configuration*. meaning that settings can be changed while the system is running. Most of the middleware products across the System z family are self-configuring.

With the System z platform, a data center does not have to upgrade the operating system whenever it adds a new machine. System z hardware will always run the current and three previous releases of z/OS. (Supported middleware is the current and one previous version.) This enables data centers to reduce the complexity in upgrading by choosing to upgrade just the hardware, and upgrade the software and middleware later.

z/OS requires no outages for configuration changes even if there is more than one image of the software running on the system. Self-configuring technologies are not meant to simplify the complexity of z/OS, but to simplify its configuration. z/OS has tools to help with the dynamic configuration of systems.

IBM intends to provide a new user interface for z/OS management that is planned to help the new generation of IT workers by automating, eliminating, and simplifying many z/OS management tasks. A new management console powered by Tivoli technology is in the works, as well as other enhancements to z/OS to make it fundamentally easier to set up and manage.

The following tools are available to help with configuring z/OS.

z/OS wizards

IBM provides Web-based wizards that assist with system customization. These wizards ask business-related questions to help customers decide which products can help them achieve their business goals.

Hardware configuration

Hardware Configuration Definition (HCD) is an integrated part of the operating system. It provides an interactive interface for configuring your channel subsystem and operating system. HCD creates the I/O Definition File (IODF). Changes can be made to the current I/O configuration without having to IPL the software or power-on-reset (POR) the hardware.

Hardware Configuration Manager (HCM) is a graphical user interface to HCD. HCM runs on a PC and interacts with HCD in a client/server relationship. The

host system is required to maintain an internal model of the IODF, but it is easier to maintain the model in a visual form. HCM enables you to navigate easily through configuration diagrams, make changes to your configuration dynamically, and manage the physical aspects of your configuration.

6.5 Self-healing technologies

Since its inception, z/OS has incorporated self-healing philosophies and reliability, availability, and serviceability features. New developments in z/OS are consistently trying to improve these features. This section presents some of the new tools that help the system heal itself.

Parallel sysplex automation

IBM Tivoli System Automation for z/OS (SA z/OS) is the leading Parallel Sysplex automation product. It provides policy-based self-healing of applications, system, and sysplex resources. For a full description of SA z/OS, see 6.8, “End-to-end enterprise systems management” on page 115.

IBM Health Checker for z/OS

IBM provides a framework for a predictive system with IBM Health Checker for z/OS.

IBM Health Checker for z/OS helps simplify and automate the identification of potential configuration problems before they affect system availability or cause outages. It compares settings and values on the active system to values and settings suggested by IBM or defined by your installation. IBM Health Checker for z/OS, a continuously running preventive tool, alerts you when critical system settings and values change. On the autonomic maturity continuum, IBM Health Checker provides predictive capabilities for z/OS.

This framework manages functions such as check registration, messaging, scheduling, command processing, logging, and reporting. This framework is provided as an open architecture in support of check writing. Installations can write their own checks to determine whether their system is configured to meet their business goals.

6.6 Self-optimizing technologies

One of the core values of z/OS is its ability to optimize the workloads running on the system to fully utilize the system resources. z/OS has a long history of self-optimization in the area of workload management and virtualization. The

virtualization of resources on z/OS means that they can be allocated wherever they are needed, not wherever they are physically connected.

Workload Manager and Intelligent Resource Director

Workload Manager provides the ability to handle mixed workloads. z/OS has a very high degree of utilization and can use system resources that would otherwise be wasted in a mixed-workload environment. Workload management on z/OS uses policies set by the installation to determine the allocation of resources. This management is highly automated and includes the scheduling of workloads, resource allocations, and responding to potential and major error.

The unique workload and self-management capabilities provided by z/OS Workload Manager (WLM) and the Intelligent Resource Director (IRD) enable z/OS to handle unpredictable workloads and meet response goals through effective use of CPU and I/O resources with minimal human intervention for setup and operation, making it the most advanced self-managing system.

For more information about WLM, see *z/OS V1R8.0 MVS Planning Workload Management,* SA22-7602-12. For more information about IRD, see *z/OS Intelligent Resource Director,* SG24-5952.

System-managed storage

Requirements for data and storage continue to grow. Although the cost of physical storage is decreasing, the cost and complexity of managing that storage continues to grow. The Data Facility Storage Management Subsystem (DFSMS) component of z/OS provides a range of automated data and space management functions to help improve disk usage and manage disk and tape storage growth with a centralized point of control with a storage administration policy. DFSMS can help reduce out-of-space failures, fragmentation, poor capacity utilization, and device-type constraints. DFSMS allows the most critical data to be on the fastest devices where and when the application needs it and it moves other data to slower, less-expensive devices. DFSMS also provides system management of tape devices. It can keep track of where the tapes are kept in a manual or robotic tape system.

DFSMS can automate and simplify data maintenance processes with policies. The processes that can be automated include backup, recovery, archiving, and disaster backup and recovery. DFSMS is also discussed in 3.3.5, “Systems Managed Storage” on page 46.

6.7 Self-protecting technologies

The isolation provided to applications by z/OS is a key differentiating factor. This isolation helps keep applications running on the same system from taking each other down. z/OS is not vulnerable to buffer overrun attacks due to this isolation. Intrusion Detection Services is an example of the new technologies that improve on the self-protecting features of z/OS.

Intrusion Detection Services

The security features of z/OS are unparalleled and one of the key features of z/OS self-protection is the Intrusion Detection Services (IDS). IDS enables the detection of attacks and the application of defense mechanisms on the z/OS server. It utilizes policies to determine what to do when different types of attacks occur. Types of events IDS can detect include scans, single packet attacks, and flooding. Actions include packet discard, connection limiting, and reporting.

z/OS includes key security features such as PKI, LDAP, Kerberos, SSL, digital certificates, and tape encryption. For more information about security on z/OS, see Chapter 4, “Security” on page 65.

6.8 End-to-end enterprise systems management

IBM Tivoli System Automation for z/OS (SA z/OS) plays an important role in building the end-to-end automation of the IBM autonomic computing initiative: It brings together the four facets of autonomic computing to provide a self-managing infrastructure for z/OS. SA z/OS can help customers manage single-process z/OS systems or Parallel Sysplex clusters, reducing the frequency and duration of incidents that have an impact on IT availability.

SA z/OS helps customers ease management, minimize costs, and maximize application availability. SA z/OS is a high-availability solution for critical business applications, providing self-healing processes and automation for I/O, processor, and system operations. System administrators define the “desired” state for the system, and the software will monitor and launch the appropriate response if the system deviates from the desired state.

SA z/OS comes with more than 40 plug-and-play automation policy modules including IMS, CICS, Tivoli Workload Scheduler, DB2, mySAP™, GDPS, and WebSphere. These policy modules are based on best practices, customer experiences and a deep knowledge of each application. Many commercial applications and z/OS components provide out-of-the-box automated message management that requires minimal effort in setup and helps ensure high-quality

installation. This powerful message policy supports different automation for different start types or message codes.

SA z/OS can be used as a base to leverage your autonomic end-to-end solution for systems management.

6.8.1 The value of SA z/OS

System automation in the z/OS environment is valuable almost to the point of being a requirement.

Parallel Sysplex application automation

One advantage of SA z/OS is that it makes Parallel Sysplex automation a reality. It manages any number of stand-alone systems and clusters from a single point of control, a Java-based GUI, by allowing for sysplex-wide grouping of resources, dependencies, and goals.

Policy-based self-healing

Another advantage of SA z/OS is policy-based self-healing. It reduces complexity, implementation time, coding, and support effort through an automation policy that can be shared or cloned across the enterprise. The benefits of this are that policies are built more easily using more cloning variables and more predefined and self-configured message rules. With SA z/OS you can monitor the system for signs of an outage condition and relieve those conditions before an outage actually occurs. SA z/OS includes performance-related information from IBM Tivoli OMEGAMON monitors in your automation processing framework. This framework enables you to proactively inform system administrators of performance issues in the system and terminate jobs or move workloads based on the appropriate action defined the policy, before the outage occurs.

Integration

SA z/OS integrates with Tivoli products including IBM Tivoli Enterprise Console® and IBM Tivoli Business Systems Manager to provide seamless automation of both system and network resources.

mySAP high-availability automation

SA z/OS provides a high availability management solution for mySAP and its related components. It combines the concepts of automation, high availability, and transparent failover.

6.8.2 Enterprise-wide systems management

IBM Tivoli System Automation for Multiplatforms uses an adapter infrastructure to integrate with SA z/OS, enabling you to effectively automate composite applications that span systems. This is a single point of high availability across z/OS, Linux, and IBM AIX®. This enables you to display aggregated and detailed status of application components, manage application components on all platforms with a single interface, and increase availability by helping you resolve cross-platform dependencies. SA z/OS in conjunction with SA for Multiplatforms enables you to establish a single team for operations and automation of z/OS, Linux, and AIX applications. This greatly simplifies problem determination and resolution.

For more information about Tivoli System Automation, see:

http://www.ibm.com/servers/eserver/zseries/software/sa

7

Consolidation

This chapter provides an overview of consolidation on the System z platform and discusses the strengths and constraints of available consolidation approaches. The discussions are grouped as follows:

- Consolidation overview
- Site consolidation
- Application consolidation
- Server consolidation
- Data consolidation

7.1 Consolidation overview

Consolidation is much more than simply replacing a lot of small servers with fewer, bigger, more powerful servers. Consolidation is about finding ways to align and manage your existing IT infrastructure to better support the business model, while establishing a flexible foundation designed to handle future requirements.

The goal is to optimize and simplify your IT infrastructure from end to end, not just the servers, but the storage, data, applications, networks, resources, and system management tools that help bring the entire infrastructure together. In addition to offering potential cost savings and improvements in efficiency, availability, and productivity, consolidation can provide a stable foundation for the rapid deployment of new initiatives as business needs continue to change.

Typically in a UNIX, Linux, or Microsoft® Windows® environment, workloads are physically partitioned across different physical servers. The distributed servers normally are sized with big white spaces to handle workload spikes. The servers generally run under very low utilization (Windows 5% - 10%, UNIX 25%-30%). These workloads are the candidates for consolidation.

7.1.1 Consolidation benefits

The major benefits of consolidation are:

- Lower total cost of ownership (TCO)

  Consolidation can help reduce complexity, establish better system management practices, optimize resource utilization, and control spiraling hardware and software licensing fees.

- Improved service levels of applications

  Consolidation can help you enable the applications that drive the integrated enterprise to deliver increased data access, higher levels of availability, and faster response times to users.

- Increased security and operational resiliency

  Consolidation can help manage the security of key information. Leveraging the industry-leading System z capabilities, you can provide a security-rich, agile, near-24x7 environment that is highly resistant to unplanned interruptions and can mitigate risk, transparently adapt to change, and scale to upswings and downswings in the market, as well as being capable of rapid recovery.

- Information as a strategic business tool

  The distributed computing model often creates islands of applications and data. Consolidating IT resources can help ensure that critical business data, and processes are accessible and shared across the enterprise.

7.1.2 Different types of consolidation

There are several different types of consolidation, which can be distinguished from different perspectives: centralization, application consolidation, data consolidation, and server consolidation.

Site consolidation

Site consolidation means moving servers from different locations to one common location. Because it simplifies access for IT staff, it helps to reduce operations support costs, improve security, and ensure uniform systems management. Figure 7-1 depicts the concept of site consolidation.

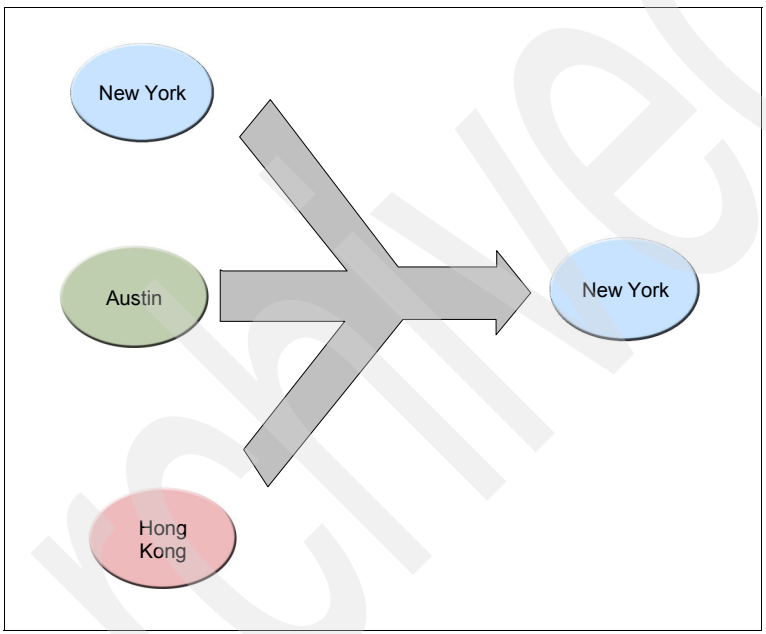


Figure 7-1 Site consolidation

Application consolidation (n-to-1)

Application consolidation means merging applications that are running in several servers into one server with one operating system image. Figure 7-2 shows the full consolidation concept.

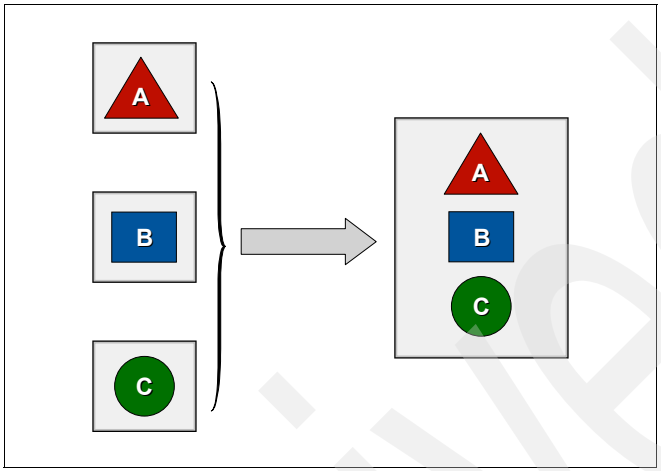


Figure 7-2 Application consolidation

Reducing the number of systems and application installations from many to one can significantly lower the expense and effort of administration and maintenance cost and improve disaster recovery. This is possible if you have an operating system that can scale up and is one of the reasons why z/OS can provide value. In addition, depending on the software licensing model, application consolidation can also save software license fees.

However, application consolidation could be time consuming and require application architecture and code changes and migrations. At times application consolidation might be very difficult or not even feasible for various reasons:

- Isolation problems:
  - A strong enough isolation might be possible only with separate operating system images.
  - Applications might not be able to live together and one application could crash the operating system.
  - Maintenance packages might conflict with each other.
  - Application changes might require operating system changes that affect other applications.
- Heterogeneous workload management and performance problems

- A software vendor who is very specific about support on applications and platforms, so the customer might risk losing support from the software vendor if the platform is changed

The complexity of application integration could offset the cost savings mentioned before.

Server consolidation (n-to-n)

Server consolidation consolidates the distributed servers into the same number of operating system images on one box by using virtualization technologies. The virtualization technologies allow sharing of hardware between multiple operating systems, leading to better utilization of hardware resource. Figure 7-3 illustrates the concept.

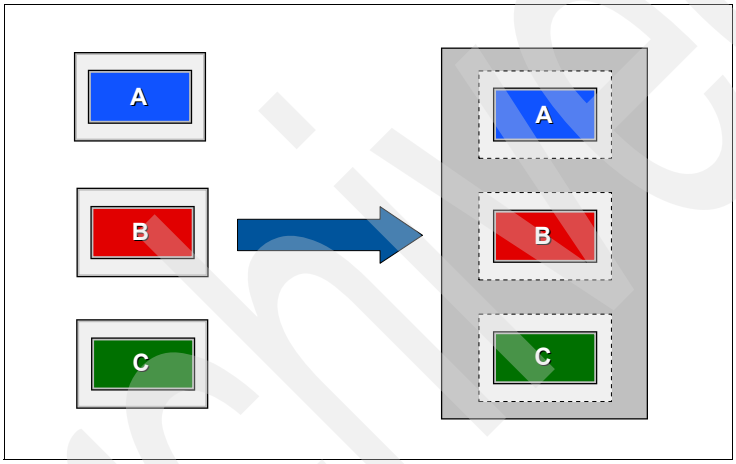


Figure 7-3 Server consolidation

Even when the number of operating system image installed is still the same, the administration and maintenance cost can still be reduced. Deployment of new servers will be faster because the new server is just a logical definition and can share the available hardware resources.

Data consolidation

Data consolidation is the process of merging data from different sources into a single repository and a common format and, at the same time, consolidating the data storage. Figure 7-4 shows the basic concept of data integration.

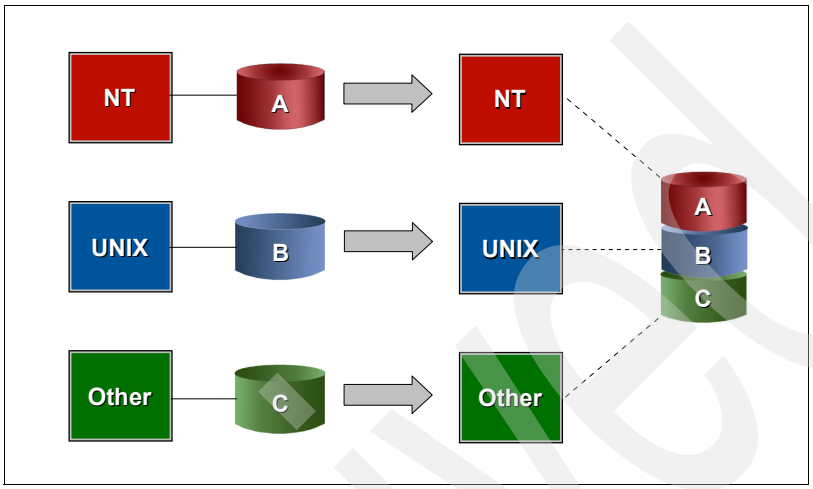


Figure 7-4 Data consolidation

When all corporate data resides on the same robust system, the efficiencies can deliver immediate payback to end users. Data sharing throughout the enterprise is vastly simplified. Consolidation enables high levels of security and data integrity.

Data consolidation is not an easy task. It requires an enterprise data architecture for the actual data consolidation. The application architects have to adjust the application architectures to accommodate the new data access layer.

The consolidation overview just described does not include a Quality of Service (QoS) discussion for consolidation. This does not mean that QoS is not important to consolidation; on the contrary, an IT architect should fully understand the QoS implication of a consolidation solution before making any consolidation-related architecture decisions.

7.1.3 Why consolidate on System z

System z has been the main platform for enterprise data processing operations for many years. The unique strengths of the System z platform (its ability to run multiple workload types with exceptionally high reliability, throughput, security and manageability) have been imitated but never matched by other server platforms.

Figure 7-5 shows the basic vision of System z consolidation: a data center in a box, not a server farm.

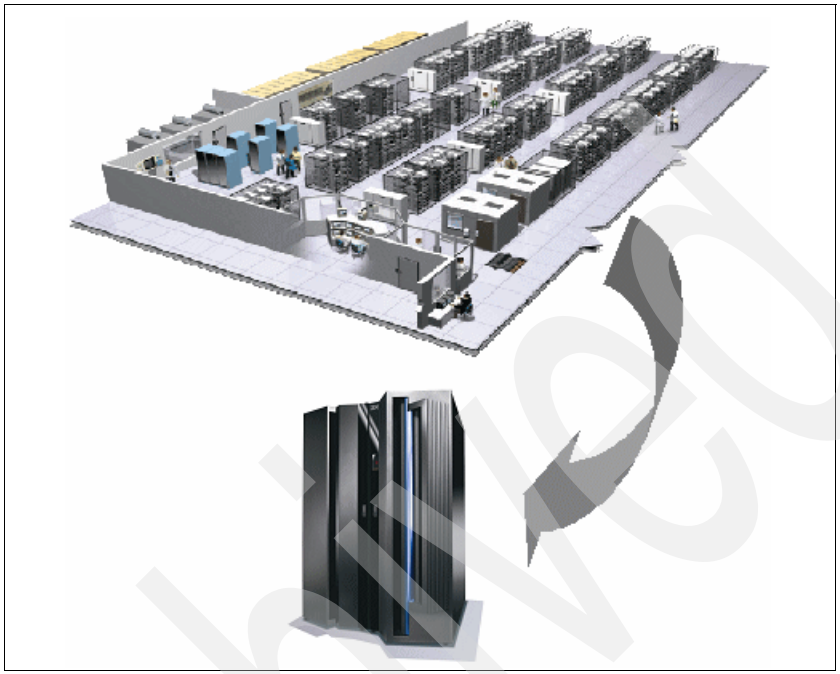


Figure 7-5 Data center in a box

The following strengths make the System z platform ideal for consolidation:

- z/Architecture is designed for handling heterogeneous workloads. The Workload Manager (WLM) and Intelligent Resource Director (IRD) manage the mixed workloads and computing resources to help applications that share everything.
- The System z platform has advanced virtualization technologies.
- The System z platform has a strong heritage in data serving. Its architecture is designed for massive data access, whether across the Internet, to storage devices, or to remote backup sites. The specialty engine zIIP makes data integration more attractive. (See “Enterprise hub for data” on page 163.)
- Building on top of WebSphere Application Server for z/OS, the System z family includes a large stack of middleware products to support new workloads for on demand business. The zAAP specialty engine makes running Java workload on z/OS cost effective. These new System z workload capabilities make application consolidations feasible.

- The System z environment has been designed to deliver the system integrity and security-rich solutions needed to help meet today's on demand operating environment security requirements. (See "Security" on page 65.)
- The System z platform provides unmatchable QoS. Consolidating to System z allows the applications to leverage its exceptional reliability, high availability, scalability, serviceability, and resiliency. (See "Resiliency" on page 97.)
- Sophisticated system monitoring and management mechanisms have been available on System z for years. (See "Systems management" on page 107.)

However, this does not mean that you should consolidate everything to the System z platform because it simply is not right for some workloads. The architecture decision is always a trade-off between the benefits and constraints.

The following sections discuss various techniques for consolidating from a distributed environment to the System z platform.

7.2 Site consolidation

Traditionally, many servers were distributed in different locations because of network bandwidth limit and network availability to the data center. Users could still work with the servers in the distributed locations when the connection to the data center was not available. Today, the network is faster, more reliable, and less expensive, so it is feasible to move distributed servers to one location.

Site consolidation is an important predecessor to other, further consolidations such as application consolidation, server consolidation, and data consolidation.

Site consolidation normally does not happen in the System z platform unless the company wants to merge their data centers.

7.3 Application consolidation

In normal situations, distributed applications can be consolidated to either of two operating systems on the System z platform:

- z/OS
- Linux on System z

This section outlines the application consolidation technology choices of z/OS and Linux on System z, discussing the major technology strengths and constraints.

7.3.1 Application consolidation to z/OS

Applications can be consolidated to one z/OS image from multiple z/OS images and distributed servers.

The applications that can be consolidated from z/OS to z/OS are developed using traditional programming languages such as COBOL, PL/I, Assembly, and others. The following discussion focuses on consolidating applications from distributed servers, but the same concepts apply to z/OS-to-z/OS application consolidation.

When we talk about consolidating distributed applications to z/OS, most likely the workloads are Java and C/C++ workloads. In the past, more C/C++ workloads were consolidated to z/OS UNIX System Service (USS). Deploying C/C++ workloads to USS is relatively straightforward. However, the USS must be sized properly to handle the workload.

Lately we see more and more Java workloads consolidated to z/OS from distributed servers.

Java workload support on the System z platform

IBM JDK™ for z/OS, WebSphere Application Server for z/OS, and zAAP are the enablers for running Java workloads on z/OS.

Figure 7-6 shows the basic architecture of WebSphere Application Server for z/OS.

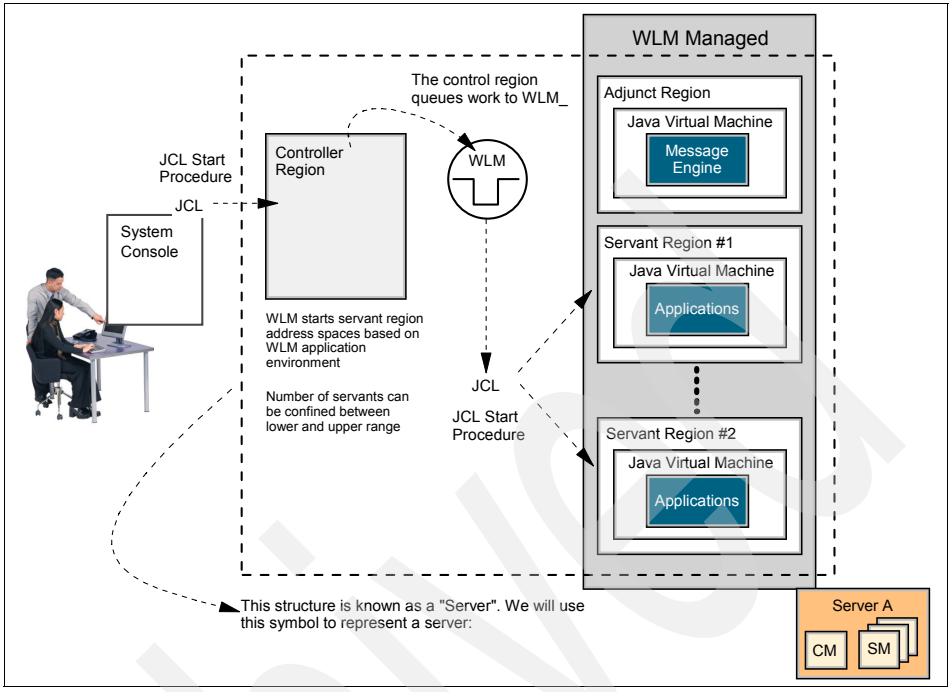


Figure 7-6 WebSphere Application Server for z/OS

WebSphere Application Server for z/OS offers a common J2EE programming model same as WebSphere Application Server for distributed platforms. With its unique multi-process architecture shown in Figure 7-6, WebSphere Application Server for z/OS provides tight integration with the System z platform to take advantage of System z strengths.

The WLM for z/OS goal-oriented infrastructure is uniformly extended to all workload types that run on z/OS. It is one of the cornerstones for consolidating Java workloads to z/OS:

- Within WebSphere Application Server for z/OS, workloads classified into different service classes are properly handled.
- Within z/OS, the WebSphere Application Server for z/OS workload is properly managed against other workloads running in the same system.

Java applications consolidation benefits

As depicted in Figure 7-7 and Figure 7-8, when a J2EE application is consolidated into WebSphere Application Server for z/OS servers, a multiple physical tiers architecture is moved into a single physical tier. However, the multiple logic tiers architecture stays without changes.

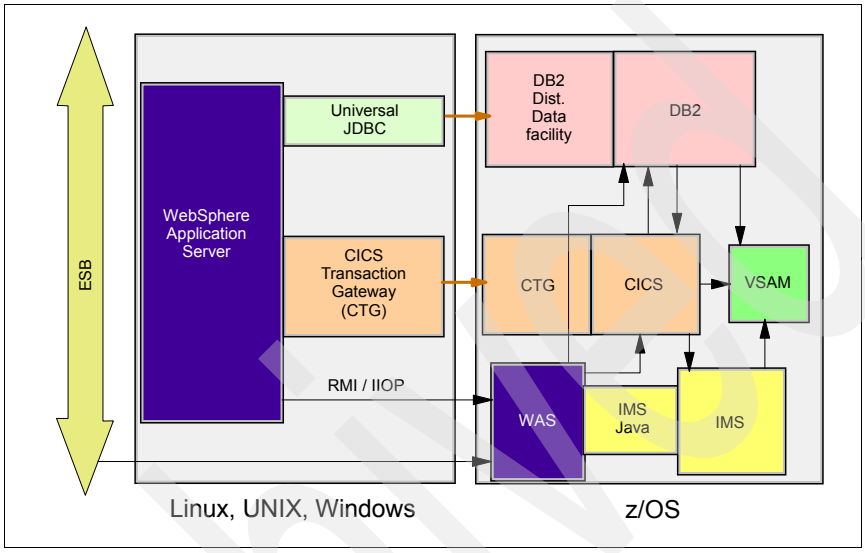


Figure 7-7 Before application consolidation

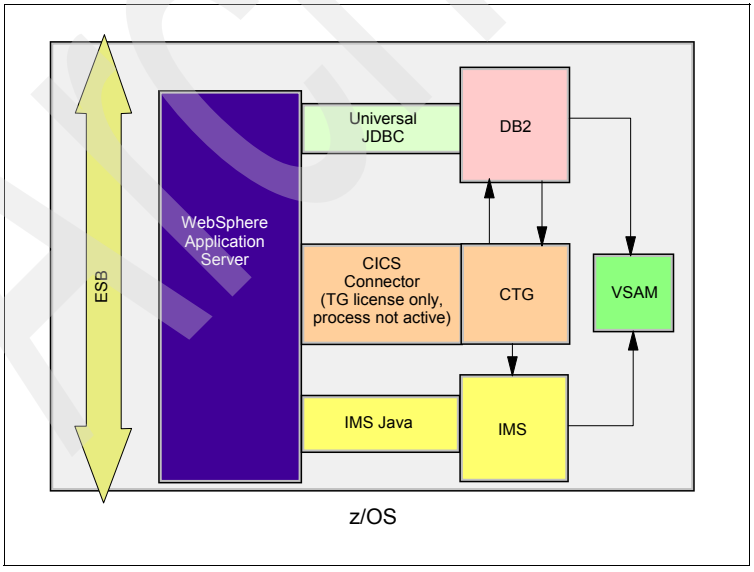


Figure 7-8 After application consolidation

This movement brings many benefits:

- Lower risk of security breach.
- Continuous availability by leveraging z/OS Parallel Sysplex technology.
- Unmatched business resiliency.
- Highest resource use efficiency/utilization.
- Resource virtualization.
- High scalability: z/OS architecture favors vertical versus horizontal scaling.
- Intelligent fine-grained workload management.
- Proximity to data: Overall transaction throughput would be improved, and the memory required to run the same workload would be reduced.

All of these benefits are gained without major application architecture and code changes.

Deployment considerations

Architecture decisions such as those that follow are trade-offs on application isolation requirements, non-functional requirements, and the backend EIS accessed by the application:

- How many WebSphere Application Server for z/OS servers (clusters) are needed for the consolidation?
- How many servant region address spaces are needed for a specific WebSphere Application Server for z/OS server?
- Where should the WebSphere Application Server for z/OS servers (clusters) be created?
- Which server should a specific application be deployed to?

To achieve high-level application isolation, you might want to deploy different applications to different servers.

The WebSphere Application Server for z/OS architecture favors heterogeneous high-volume applications. WLM starts and stops the WebSphere Application Server for z/OS server servant regions to handle the dynamic workloads, so the resource utilization is more efficient. However, when you consolidate a large number of high-volume applications to WebSphere Application Server for z/OS to a single z/OS image, the single z/OS image might not have enough power to handle the workload.

Deploying a large number of low-volume applications to WebSphere Application Server for z/OS is not a wise decision to make. It is hard to justify the cost unless these applications are business mission critical and need the QoS of z/OS.

It is a good idea to make the WebSphere Application Server for z/OS server co-locating with backend data stores and EIS resources on the same z/OS image. The application can have better performance and better security.

Application performance

Performance engineering in a shared environment is harder than that in a physical isolated environment. In general, running applications in WebSphere Application Server for z/OS can achieve performance similar to running applications in WebSphere Application Server for distributed environments.

The performance engineering includes z/OS tuning, WebSphere Application Server for z/OS server tuning, and subsystem (such as DB2, CICS, IMS, or MQ) tuning.

Be aware that all workloads on z/OS are under WLM control. Running applications in WebSphere Application Server for z/OS is resource intensive. An aggressive but reasonable performance goal is needed to achieve desired performance.

Middleware and third-party software

The middleware choice is limited to IBM products, to some extent. If you consolidate some applications using non-IBM middleware, most likely you have to migrate to an IBM alternative for the consolidation. The application architecture and code changes are necessary in these situations.

Some third-party software is not supported on z/OS, which might present a big challenge for some applications.

Summary

The major strengths for consolidating applications to z/OS are the unmatched QoS, heterogeneous workload management, and high resource utilization. However, the candidate workloads are somehow limited to Java/C++ applications. Performance engineering is more complicated than on a distributed environment. There is less freedom to pick and choose middleware and third-party software.

7.3.2 Application consolidation to Linux on System z

Linux on System z is able to run in three modes: basic, LPAR, and z/VM guest. The most common way to consolidate distributed applications is to a single image of Linux running in one of the other two modes, shown in Figure 7-9, depending on applications footprint and resource usage.

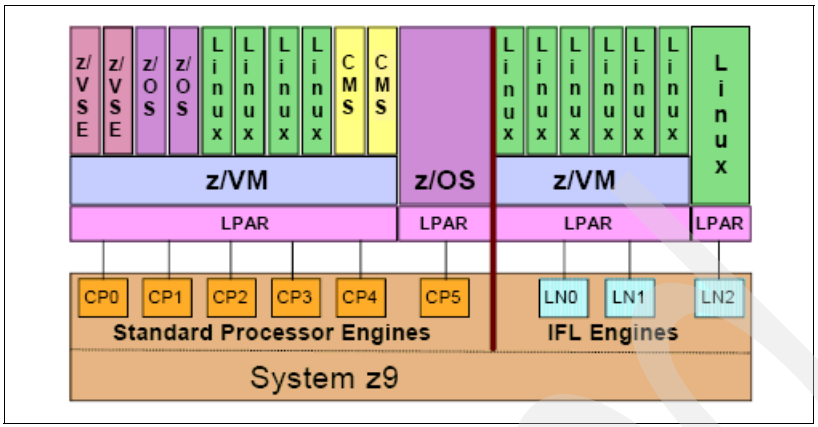


Figure 7-9 Linux on System z

Virtually any application that is portable to Linux can be consolidated to Linux on System z.

Leverage the classic strength of System z

Linux on System z takes advantage of the inherent value of z/Architecture:

- Extremely high availability of the hardware platform
- High I/O bandwidth capabilities
- Flexibility to run disparate workloads concurrently
- Excellent disaster recovery capability
- Security and resiliency

However, Linux on System z does not provide all of the features that are available on z/OS. The application isolation in a single Linux image on System z is not as good as that in a single z/OS image. At this time, no WLM equivalent component exists in Linux on System z to handle application workloads. So the single Linux image on System z cannot handle the heterogeneous workloads as well as z/OS.

Business integration

HiperSockets enables LPAR-to-LPAR communication within a System z server at memory speed. It shortens the end-to-end application path length for the applications that are accessing EIS resources on z/OS.

System z servers allow running multiple Linux, z/OS, z/OSe, z/VM, TPF, and VSE/ESA™ systems at the same time as shown in Figure 7-9, along with diverse middleware components. The result is a highly integrated environment in which the consolidated applications can share information and be more responsive, flexible, and efficient.

Application performance

Again, a shared environment has more challenges for performance engineering. A system tuned for one type of workload could perform poorly for another type of workload. It is important to understand the application workload and be prepared to review system tuning when the workload changes. See *Linux on IBM zSeries and S/390: Performance Measurement and Tuning,* SG24-6926 for more information.

Middleware and third-party software

Built on open standards, Linux on System z gives you the freedom to choose middleware. For example, a distributed application using non-IBM middleware can be consolidated to a Linux on System z image without migrating to IBM middleware.

Linux on System z supports a lot of third-party software and open source products. The software licensing model for Linux on System z also might offer a cost advantage.

Summary

Compared to application consolidation with z/OS, Linux strengths include less workload limit (not just Java, C++) and the freedom to choose middleware and third-party software. Applications consolidated to a single Linux image on System z also have short EIS resource access paths. The performance engineering is complicated as well.

The QoS that Linux on System z can provide is lower than that from z/OS. The application isolation in a single Linux image on System z is not as good as in a single z/OS image. It might not be a good idea to consolidate different types of workloads into a single Linux on System z image because Linux on System z does not have a mechanism similar to z/OS Workload Manager.

7.3.3 Application management

The System z platform has sophisticated workload monitoring and management facilities. Besides built-in facilities such as RMF and SMF, there is a family of IBM Tivoli products that conduct application management in the System z environment.

The IBM Tivoli Composite Application Manager family resides in the application management pillar of the Tivoli software portfolio. The current application management portfolio consists of the following products:

- IBM Tivoli Composite Application Manager for Response Time Tracking V6.0
- IBM Tivoli Composite Application Manager for SOA V6.0
- IBM Tivoli Composite Application Manager for WebSphere V6.0
- IBM Tivoli Composite Application Manager for CICS Transactions V6.0

- IBM Tivoli Composite Application Manager for IMS Transactions V6.0
- IBM Tivoli OMEGAMON XE for WebSphere Business Integration V1.1

These products provide a comprehensive solution for applications running on both z/OS and Linux on System z.

7.4 Server consolidation

The major technology behind server consolidation is virtualization. The multiple distributed servers are consolidated into the same number of virtual servers on one box.

The System z platform has two possible server consolidation architectures in most situations:

- Consolidating the distributed servers to same number of LPARs. Each LPAR has one operating system image: either z/OS or Linux on System z.
- Consolidating the distributed servers to the same number of Linux images running as z/VM guests.

This section discusses the technology strengths and constraints for these two server consolidation architectures.

7.4.1 Server consolidation using LPARs

Processor Resource/System Manager (PR/SM) is a type 1 hypervisor integrated into all System z models that transforms physical resources into virtual resources so that many LPARs can share the same physical resources.

PR/SM provides the ability to divide physical system resources (dedicated or shared) into isolated LPARs. Each LPAR operates like an independent system running its own operating environment. On the latest System z models, you can create up to 60 LPARs running various System z operating system images on a single system.

Server consolidation to z/OS LPARs applies only to consolidating multiple z/OS images running on different physical boxes. If you try to consolidate distributed servers to z/OS, then workloads are somehow limited to Java workloads. The consolidation is not transparent to application developers: application migration might be triggered because the middleware that is used by the application is not available on z/OS. Therefore it is no longer a server consolidation but an application consolidation mentioned in 7.3.

In theory, it is possible to consolidate distributed servers to Linux images running in LPAR mode. Plenty of software supports on Linux on System z makes the server consolidation fairly transparent to application development folks. Unfortunately it is hard to justify using a System z LPAR to replace a low-utilized distributed server unless you are trying to consolidate some fairly big distributed servers that have big footprints.

7.4.2 Server consolidation using z/VM guests

The most flexible and resource-efficient way to run Linux on System z is through z/VM guests. Linux running as z/VM guests is perfect for consolidating the distributed servers with low utilization.

z/VM and Linux on System z

z/VM is the System z virtualization technology. As a type 1 hypervisor, z/VM in concert with System z hardware provides virtualization of CPU processors, I/O subsystems, and the memory.

The z/VM control program (CP) virtualizes the hardware resource either by partitioning or sharing real hardware resources, or by emulating their behavior. The runtime, which CP offers to the guest operating systems, is called the *virtual machine*. It has its own memory, devices, and processors, which can be either real or virtual. Tens to hundreds of Linux servers can be created as z/VM guests.

A special z/VM disk type is the *minidisk*. Minidisks are virtual disk devices; they can be either a part of or a complete DASD. Minidisks can be shared among several guest systems. A real disk can also be dedicated to a virtual machine.

The communication between the guest systems and z/VM can be established with various virtual networking provided as part of System z virtual technologies.

Consolidation strengths

Linux images running as z/VM guests form a virtual distributed environment. A large number of distributed servers can be consolidated in this virtual distributed environment. Figure 7-10 on page 136 shows one of the server consolidation examples.

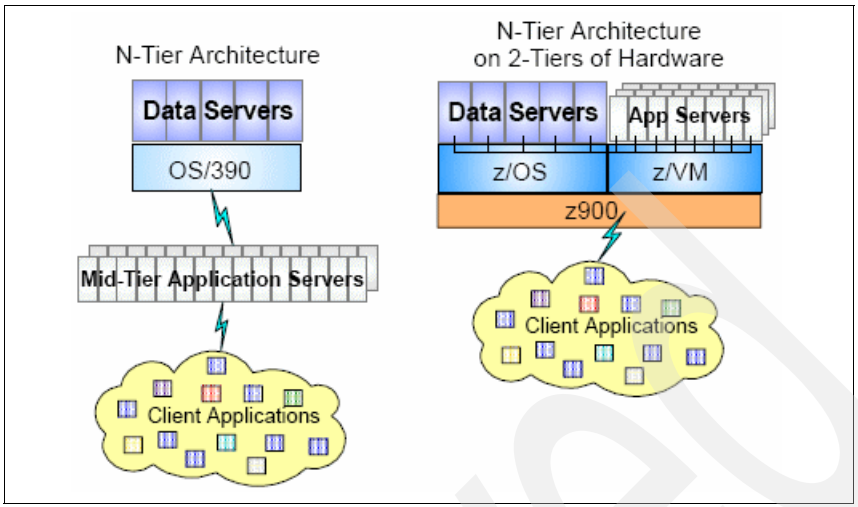


Figure 7-10 Application servers consolidation

There are a lot of benefits from this server consolidation: Besides the cost savings from server farm elimination, management is easier. The software licensing model of Linux on z/VM has the potential for additional cost savings.

The virtual networking technology reduces the end-to-end application execution path and improves the security risk. Linux on System z solutions can also incorporate several industry-leading security features such as enhanced cryptography support designed to keep online transactions private and secure.

System z servers with z/VM can help you deal with unexpected increases in workload. You can bring Linux servers online in a matter of seconds or minutes with z/VM. You can preconfigure Linux servers offline (that is, logged off) and bring them online as soon as you need them. It is also possible to provision the server in real time in a matter of minutes to help minimize costly transaction delays and potentially devastating system crashes. Virtual machines are easy to create, provision, modify, and decommission when you no longer need them.

The z/VM guest servers on System z are isolated, so you can run multiple heterogeneous workloads simultaneously.

Constraints

Linux on System z does not provide the same level of QoS as z/OS. The workload management is not as sophisticated as zWLM. Linux on System z does not have the same capability as z/OS to handle very high volume transactions.

A selection of third-party software and open source products is available for Linux on System z, although several ISV software packages are not yet available for Linux on System z. A server consolidation to Linux on System z is possible only if software is available or migration to an available alternative is acceptable.

Suitable workloads

Workloads with the following characteristics are suitable for server consolidation to Linux on z/VM environment are:

- Workload can benefit from being physically close to System z data.
- Workload with high I/O.
- Workload requires high availability.
- Workload requires excellent disaster recovery.
- Workload requires better security.
- Servers have low to medium utilization.

Therefore servers running the following products are considered as best fits:

- IBM WebSphere MQ
- IBM DB2 Connect™
- IBM CICS Transaction Gateway
- IBM IMS Connect for Java
- IBM WebSphere/WebLogic and Java application development
- Applications requiring top-end disaster recovery model
- ComServer and Communication Controller for Linux
- LDAP security services
- IBI Web Focus
- Oracle® 10g DB

Servers running the following products or workloads are considered as good fits:

- IBM DB2 9
- IBM Informix® Dynamic Server (IDS)
- WebSphere applications
- Apache Web Serving
- Samba
- SAP
- Network infrastructure, firewall, FTP, NFS, DNS, and so forth
- e-mail solutions

Most WebSphere applications require DB2 as database. More and more ISV products are supporting DB2 under Linux on System z as z/VM guests. If high quality of service is required, you can consider using DB2 for z/OS with the benefits of the data-sharing environment.

If QoS provided by DB2 data sharing and sysplex are required, you should consider moving to DB2 for z/OS. Then applications will access DB2 through either DB2 Connect or a Type 4 JDBC™ driver.

WebSphere applications running under Linux on System z on z/VM keep the distributed architecture model, but they have the advantage of being close to legacy data.

Distributed infrastructure's scale out architecture is good at handling a large number of simpler applications. Running Linux on z/VM creates a similar virtual environment with the inherent System z classic strengths. It is easy for the Linux on z/VM environment to handle a large number of servers running similar applications.

7.4.3 System management

IBM provides sophisticated tools and products for the Linux on System z environment. For more information, refer to Chapter 6, "Systems management" on page 107.

7.5 Data consolidation

The System z platform has a strong heritage in data serving, and its architecture is designed for massive data access, whether across the Internet, to storage devices, or to remote backup sites. Scalability, availability, security are all core competencies of the System z environment. Its role is enhanced as the data hub for the enterprise. It helps merge Transaction Processing and Business Intelligence. Refer to Chapter 9, "Enterprise hub for data" on page 163 for information about data consolidation in System z as an enterprise data hub.

A common data access layer should be introduced in enterprise application architecture to decouple applications from data consolidation. For example, in SOA context, the common data access layer most likely is exposed as the information services. The IBM WebSphere Information Integrator family of products and IBM WebSphere Transformation Extender (WTX) products can be leveraged to provide the universal data access services.

7.6 Choosing the right consolidation approach

For the most part, consolidation enables administrators to gain a single point of control while maintaining flexibility and the ability to prioritize service levels for end users. However, one of the most important things to remember about

consolidation is that there is no off-the-shelf solution or one-size-fits-all answer for IT optimization.

In general, z/OS is more suitable for application consolidation, especially for consolidating a smaller number of applications with high transaction volume or having high QoS requirements. Linux on System z is capable of consolidating a large number of underutilized distributed servers. Each approach leverages classic System z strengths in a different way.

8

Enterprise hub for SOA: integrating and extending assets

In the previous chapters, we presented the qualities that make the System z platform unique. In this chapter, we discuss the role of the System z platform in a service-oriented architecture (SOA).

We start with a discussion about how IT can bring more benefits to business. We present the objectives of SOA and how this architecture offers new ways to address integration projects.

Then we detail how the System z platform can participate in an SOA environment and present the IBM SOA tools that are available on z/OS.

Finally, we highlight how the technical strengths of the platform provide true business benefits in an SOA context.

8.1 Rebuilding IT on new principles

In the past 40 years, IT has been very successful in helping businesses and organizations grow and become more efficient.

However, this very success has created a new set of issues. IT is about automating processes, and automated processes can lead to inflexibility.

The very concept of IT application is becoming more difficult to define as more efforts today are, above all, about integrating existing IT assets with new functionalities and less about creating completely new usages.

Several industries have experienced new concepts that, if adapted to the IT trade, could bring a lot of new benefits.

For a long time, the automotive companies have created specific parts and plants for each model of car they made. At the end of the 1970s, the idea to build new car models from a “bank of parts” has started to change radically the way new car models are designed. Some parts such as chassis have been made common for several models and some car parts have been entirely subcontracted to a network of component manufacturers.

The personal computer industry has experienced similar trends. Standard parts such as processors, memory, and video adapters are built by specialized companies. PC companies only define the architecture of the machines they want to sell. They have even stopped assembling the parts that compose their machines and have subcontracted that assembly work too.

In both cases, the ability to ship new products more quickly has largely improved. It takes less than three years to launch a new car model, where it used to take 10. Personal computers evolve so quickly now that it is difficult to keep pace with the new offerings. At the same time, it has helped to keep their prices low.

The software IT industry as a whole would love to reap the same kind of benefits: being able to keep the benefits of automation as well as the low cost and flexibility associated with assembly and reuse. In the software IT industry, the parts and components have been named *services*. The technical efforts to reshape the industry around these new principles are known as service-oriented architecture.

Building software systems from existing pieces is not new, and several attempts have been made in the past to go along that path. They have mostly stayed in the technical area. They were more about code reuse than sharing active business services. Those previous attempts have shown limited benefits because the various pieces were not truly able to work together. It is only recently that true interoperable standards have started to be commonplace with the advent of the

Internet and its base protocols: presentation through standard browsers, HTTP for information exchange, TCP/IP networks, and XML description of data. The advent of the for example, standards have enhanced business interoperability mechanisms that were lacking previously.

Building cars and computers is different from building software application assets. New cars share roads and gas stations with older cars. New computers can share desk space with older ones. Large software systems are much more complicated, and the migration to a design by assembly must take the existing assets into account. In that respect, software systems are closer to urban renewal, where new boroughs have to be built alongside existing ones while sharing facilities such as roads, public transportation systems, water and power networks, education facilities, and all of the services needed to ensure quality of life.

The large investments that have been made on IT capabilities now impede nimbleness and efficiency. Past investment allows the day-to-day operations of companies and government agencies but prevents them from being reactive enough to changes in their environment.

These past investments created assets that must be maintained and kept current. As the stock of IT applications and programs increases, the maintenance costs get larger.

This situation is not specific to IT:

- In the construction industry, the upkeep of a large building over its life become a significant cost that can be larger than the initial investment.
- In the aerospace industry, the initial cost of an airplane or a helicopter is covered several times by the costs of its maintenance over the 20 to 30 years of its life.

Most of the efforts of an established IT shop, and most of its budget, are spent on allowing existing applications and the business processes they underpin to run as is and achieve some integration between existing assets. Progressively, the IT departments are no longer aligned with their business sponsors. With current practices, they just cannot keep up with two opposite aims at the same time: maintenance and innovation.

This assessment leads to the need for a major shift in the way the IT industry should offer its services to businesses. This shift in mindset is based on the principles detailed in this section.

Figure 8-1 on page 144 illustrates the previous comments. To overcome the “heritage” effect, IT organization should modify the way they build software business systems. SOA offers an architecture to address those needs.

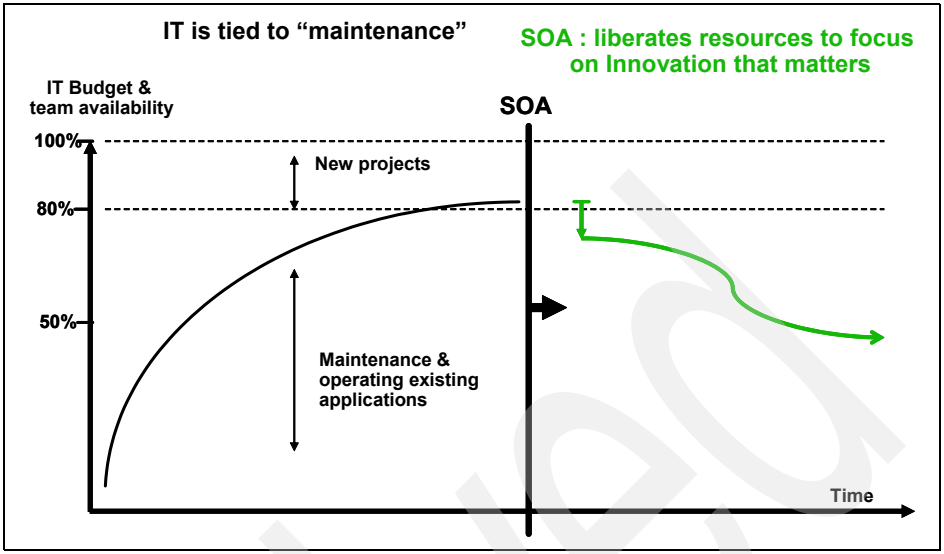


Figure 8-1 What is SOA about?

The following sections discuss key concepts that guide a service-oriented architecture.

8.1.1 Shared versus specialized components

Traditional applications were built to solve a particular issue: to take orders from customers, to manage withdrawals from ATMs, to enroll students on a campus, and so forth.

The key characteristics of those applications is that they are specialized and rarely shared components with other applications. The main reason is simple: specialized applications are more efficient in the short term.

In the current business context, value comes mostly through integration. The ordering system must be linked to the production system and the billing system and the service system and others. It begins to make sense to think at a global application level and envision the sharing of operational business services. The term “operational” is key here. SOA addresses this aspect by being based on common active services instead of duplication of reusable services serving separate applications.

8.1.2 Assembly versus build

You might need to adjust your ideas about what an application stands to understand what SOA is about.

Usually an application is a package of items, all of which are important to achieving the application's goal but only a fraction of which are really tied to specific business needs.

- Presentation logic: What the user sees, also known as the user interface or user interaction logic.
- Process logic: Handles the way the various actions are sequenced and helps the user navigate the process embodied in the application. For example, It deals with linking the "parts to be ordered" screen choice with the "customer shipment details" view. Business rules are often embedded in this process code: for example, to validate an order, you might need to check the customer's credit or solvency.
- Business function: The real business benefit for the company. It is the code that actually prices an order, creates a customer record, or performs some other significant function for the company.
- Data storage logic: Handles the manipulation of data so that it can be stored and retrieved from files or databases.
- Integration logic: The layer that has grown faster in recent years as companies have tried to make applications communicate with each other despite a lack of interoperability standards. It deal with the following items:
  - Events: What events must be communicated from one application to another?
  - Destinations: How do you find the technical details about where the target application is?
  - Communications: What network channel should be used to communicate from one application to another?
  - Data transformation: How should data format differences be taken into account? If a company has a CRM system based on Siebel® and an ordering system based on SAP, how do you reconcile the customer numbers if they are different?
- Management and monitoring logic: Often a large part of an application. There is a programmer's statement that says that most application code deals with potential errors and not with a specific business function. The application must be monitored as well and produce statistics about what it has achieved to help administrators keep track of the way the application is running.

- Security logic: The application must take care to know who is accessing it and for what purpose. It should identify users and grant them access only to the business functions they are entitled to use.

For all of these pieces, there are ways to share components: a portal might handle the presentation logic, a process server might be responsible for the choreography of tasks, an integration server might be responsible for handling the communication aspects between applications, or a security server might handle the security tasks.

When this infrastructure is achieved, the focus of application development changes radically from build to assemble. Of course, some new functionalities will still imply that new code is written. However, most of the various parts or services that constitute an application will be available. Their assembly or wiring together will enable the creation of a new business application at a fraction of the time and cost of the traditional application development efforts.

8.1.3 Enterprise vision versus project vision

The new environment described in the previous sections works best when the potentially reusable services and the shared infrastructure have been defined. That is typically not within the scope of a single project.

SOA is at its best when it is the basis of an enterprise vision and not a project-by-project vision.

In a world of shared services, the definition and upkeep of a registry of available services is an enterprise mission. The costs of shared functions should be accounted for at the enterprise level and not at the individual project level. When a service that was developed for a specific business function is relevant to another, the additional costs incurred to run this service for a larger scope, including potential new processing capabilities and new software licenses, should be shared adequately and not only by the service provider. Another key topic is change management, which becomes a key success factor when services start being used by multiple projects.

The current IT governance objectives are still necessary:

- Establishing decision-making rights associated with IT
- Establishing mechanisms and policies that are used to measure and control the way IT decisions are made and carried out

SOA builds on this existing IT governance model. It is focused on the life cycle of services to ensure the business value of SOA. A steering committee consisting of technical and business representatives must detail what the business expects

from the IT system: new offerings, new efficiencies, adaptability, and so forth. A technical committee should define the enterprise SOA standards and enforce them at the enterprise level.

8.2 SOA principles

SOA has multiple facets. It can be approached differently according to organizational roles:

- The business side sees SOA as an enabler for business flexibility. SOA is a base on which they build a set of capabilities that their organization wants to expose to clients and partner organizations.
- Enterprise architects view SOA as an architectural style. It addresses characteristics such as loose coupling and reuse.
- Developers think of it as a programming model complete with standards, tools, methods, and technologies such as Web services.
- The people in charge of operations see a set of agreements among service requestors and providers that specify the quality of service and identify key IT metrics.

8.2.1 What is a service?

Before going further, we must define service within an SOA more precisely. From an architectural point of view, SOA is a software system architecture style in which applications are built from components called *services*. A service can be defined as a repeatable business task, such as opening a customer account or pricing an order. SOA is about linking business services together to enable business processes.

A service either provides information or facilitates a change to business data from one valid and consistent state to another. The implementation of a particular service should have no importance as long as it responds in the expected way to its invocation and offers the required quality of service.

Services have well-defined interfaces and are loosely coupled in order to provide flexibility, interoperability, and reuse. They are called by a service requestor.

The most important characteristics of a service are:

- Reusability. Services must be envisioned as reusable by definition. Reuse can have two meanings:
  - New ways to use existing pieces of code or programs instead of writing new ones.

– Common use or sharing. We tend to favor this one because we think that SOA above all is about the sharing of active business services.

Not every piece of code should be called service because not every piece of code has value in being reused in a context different from the one it has been created for. Finding the right services is a key challenge of an SOA design: SOA is not about destroying and putting into pieces applications that work fine today. It is about finding or creating key technical assets that underpin shareable business functions. It could seem like overkill to add “service qualities” to a granular software function that has absolutely no chance to be reused. The emphasis on reuse is a key driver of the SOA business benefits.

- Loose coupling: Services hide implementation details and minimize dependencies with the requesting party, where dependencies are the respect of agreed standards. A service is defined by the *results* it provides, not by the way it provides them. Services provided by different technical platforms can participate in providing the IT resources needed for a business goal. Loose coupling enables modification of the sequence in which services are called and is a key enabler for flexibility.
- Stateless: The status of a service should not be dependent on the result of the last call it served. Services neither remember the last thing they were asked to do nor care what the next is. Services are not dependent on the context or state of other services.

The SOA infrastructure powers the use of different services wired together.

Within an SOA the execution engine is separated from the flow engine, whereas traditional applications embed the two functions. This specificity is the key point that allows the modification of the flow logic between services without affecting the execution of the services themselves.

8.2.2 The SOA life cycle

SOA puts a lot of emphasis on the creation of strong links between business needs and IT developments. Every technical effort has to respond to a simple question: What business requirement does it answer?

It is possible to think of SOA in terms of a virtual circle, also known as the SOA life cycle, as shown in Figure 8-2 on page 149. It has four major steps:

1. *Model:* Business analysts describe their business process requirements using modeling tools such as IBM WebSphere Business Modeler. Modeling is coupled with simulation tools to validate and optimize the behavior of the processes with various sets of parameters.
2. *Assemble:* The models created by the business analysts are shared with the IT assemblers. These are the people who wire together existing services (and

ask software developers to create new ones if needed) to satisfy the requirements using tools such as IBM WebSphere Integration Developer.

3. *Deploy:* The assembled “artifacts” are ready to be placed on the execution engines, which will deliver the requested business function. In a secure and integrated environment, they take advantage of infrastructure services that provide support for integrating people, processes, and information.
4. *Manage:* After deployment, customers manage and monitor the composite applications and underlying resources from both an IT and a business perspective. Information gathered during the Manage phase is used to gain real-time insight into business processes, enabling better business decisions and feeding information back into the life cycle for continuous process improvement. The flow engine is able to record of all the major steps of the ongoing processes. Tools such as IBM WebSphere Business Monitor provide the business information requested by the business analysts. It helps answer such questions as how many new bank loans requests were processed today or how long it takes to process a loan. This feedback is then compared to the business goals, and actions are taken accordingly. These actions might be taken at a technical level if bottlenecks such as resource shortages appear in the handling of the process, or the organizational level if the process steps must be adapted to a newer environment.

Underlying all of these lifecycle stages is governance, which provides guidance and oversight for the SOA project.

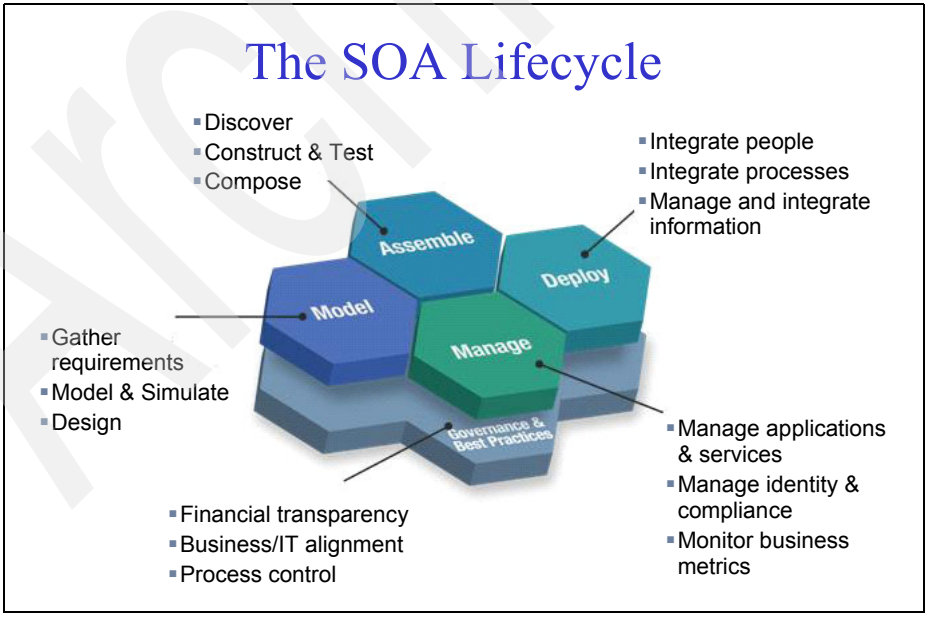


Figure 8-2 The SOA lifecycle

8.3 System z platform SOA capabilities

The SOA architecture is concerned with helping IT provide meaningful answers to satisfy business requests. It is based on business services that can run on any server platform.

Are the qualities of the System z platform relevant in that context?

This section discusses how the System z platform can participate in the SOA environment, then highlights the benefits it can offer to deliver the benefits expected from SOA.

8.3.1 A simple starting point: modernizing existing applications

Many business services are currently provided by existing applications running inside transactional monitors such as CICS or IMS.

Several points are often raised to highlight that these applications fit badly in a Web-based environment:

- They request the use of a special program to be installed on the user workstation, called an emulator.
- They use green screens instead of the graphical user interface of Web applications.
- The navigation between the screens is not always well adapted to the current business needs.

These remarks are true but can be mitigated with minimum effort:

- Emulator: Instead of installing an emulator on every workstation, a Java emulator program can be downloaded when needed from an application server into a Web browser. This eases the deployment of the application to the people who need to use it. IBM WebSphere Host On Demand (HOD) provides this functionality.
- Green screens: The green screens of classical applications can be modified on the fly and displayed in a Web-like manner in Web browsers. IBM WebSphere Host Transformation Services (HATS) modernizes screens on the fly.
- Screen navigation: The simple modernizing of classical screens might not be sufficient to provide users with the navigation they need. Traditional screens have been based on technological limits that are no longer a constraint. For example, in response to an inquiry, result sets are not sent in a single step but instead upon user request for two reasons: the old limitation to display more than 24 lines and the desire to avoid sending too much data over the network. Now scroll boxes enable a new way to design displays. A sequence of

requests can be executed simultaneously so that all answers are retrieved in a single step and made available to the user. It is possible to combine multiple other requests in a single step as well. This allows the display of more related information, about a customer for example, on a single screen: customer details, ongoing orders, and so forth. The simplification of the navigation between screens simplifies the training of application users and increases their efficiency. WebSphere Host Transformation Services (HATS) modifies the screen navigation of a traditional 3270 application, in addition to the simple screen revamping.

Service Flow Modeler

Service Flow Modeler enables business service integration for CICS applications. It provides graphical modeling workspaces so that enterprise solution architects, integration developers, and CICS application specialists who need to implement SOA can create business services that may be invoked as for example,, by composing a sequence of CICS applications.

A generator is provided that transforms the composed flow of CICS application interactions to form a CICS business service executable that may be deployed to CICS Transaction Server V3.1.

Service Flow Modeler is a multifunctional tool that supports modern application architectures and the transformation and reuse of existing application processes. It is designed to take advantage of the quality of service (QoS) of existing Enterprise Information Systems (EIS) and the CICS Runtime and Host Access Transformation Services (HATS). At the same time, it enables the move toward SOA.

The CICS Service Flow Feature, an optional feature of CICS Transaction Server V3, delivers the runtime component that is required by the Modeler to compose CICS applications.

The CICS Service Flow runtime adapters provide for access to existing CICS transaction and application interfaces using non-invasive techniques so that the CICS application assets, orchestrated by the service flow, do not have to be altered to support the CICS business service flow. This enables the fast reuse of existing assets while minimizing the risk of the new implementation. If required, the CICS business service can persist state data related to the business service between multiple invocations of the CICS application interfaces. Where there is a requirement to integrate information from non-CICS applications into the CICS business service, the CICS Service Flow runtime provides a WebSphere MQ adapter that enables the CICS business service flow to access any MQ-enabled application wherever it might reside in the enterprise solution architecture.

These capabilities are very useful to reuse existing applications and give them a new life at a fraction of the cost of the development of new ones from scratch.

8.3.2 Web services enablement

Web services are a set of open standards protocols that enable programs on different technical platforms to call each other seamlessly. They are used for integrating business processes end to end across the company and with key partners, enabling them to respond flexibly and rapidly to new circumstances. Web services enable business functions to be hosted in the environments that are most appropriate to their requirements, independent of hardware, run-time environment, and programming language.

Base Web services protocols allow cross-platform remote invocation based on the SOAP over HTTP protocol. The base Web services standards have been enhanced to cover all the functions needed for an enterprise system: service orchestration, security, transaction management, and so forth.

The key Web services standard as regards SOA is the Web Services Description Language (WSDL). It enables a standard description of services, which is key to define how to call a service and what to expect from it.

An SOA architecture implementation frequently uses Web services but is not limited to that standard.

Can the System z platform “talk” Web services? Yes it can, and in many ways:

- CICS transactions can be turned into Web services. For example, the Web services support in CICS Transaction Server V3.1 enables your CICS programs to be Web service providers and requesters. CICS supports a number of specifications including SOAP Version 1.1 and Version 1.2, and Web services distributed transactions (WS-Atomic Transaction).
- IMS transactions can be turned into Web services as well. IMS MFS Web services support enables you to reuse existing MFS-based IMS business logic as Web services in WebSphere. The MFS Web services tooling component, also known as the MFS Importer, is part of the IBM WebSphere Integration Developer. The corresponding run-time component of MFS Web services is included in IMS JDBC Connector on both distributed and z/OS platforms.
- DB2 for z/OS can act as a Web services provider to a Web services client that may run in a J2EE server. It requires that the Web services Object Runtime Framework (WORF), also known as Document Access Definition Extension (DADx), is installed. DB2 as a Web services consumer requires that you install the DB2 XML extender and that you enable Information Management Web services consumer user-defined formats.
- WebSphere for z/OS is able to run Web services that are accessible to other environments that need to access them. WebSphere for z/OS applications can call Web services that run outside the platform.

Web services alone do not define an SOA environment. If an application calls Web services and still keeps inside itself the sequence flow logic between the services, it cannot be considered to provide the flexibility that is desired in an SOA. The sequence flow logic should be external to the application so that it can be changed with the smallest impact.

8.3.3 SOA infrastructure on System z

In the previous section, we have seen that the System z platform was able to provide and consume Web services. It can do more than that: it can provide part or even the entire infrastructure needed to build an SOA runtime environment.

What does an SOA runtime environment consist of? We highlight three important servers:

- The presentation server
- The service bus server
- The sequence flow logic server

A server, in this case, can be viewed as a logical component and the listed servers can be deployed in three separate servers as well as in a single physical server.

Presentation server

In the IBM SOA reference architecture, the presentation server or interaction server is the place where the user display is aggregated. The name more commonly used is portal server. IBM WebSphere Portal Enable is the IBM offering for this function. It is available on z/OS. The portal manages small applications called portlets. Each portlet represents a business function that might otherwise have been buried inside a monolithic application. In the Portal context, an application is composed of a set of portlets. For example: the administrative functions related to customers (create, update, delete), the order entry functions, order pricing, a solvency check against an external server, and so forth. Every portal user, depending on role, may be granted access to all or some or the portlets. And of course, portlets can be shared and used in more than one application. A portal is the front end of an SOA environment that gives flexibility at the display level.

Service bus server

In an SOA environment, the business functions known as services need to communicate with one another in order to provide a full business process. Having those services communicate directly would be very impractical: It would multiply the communication channels and create a lot of administrative work in case a service is replaced by another. The SOA environment bases the

communication between services on an Enterprise Service Bus or ESB. Besides handling transport, this bus is in charge of some common mediations, such as format conversion, send to multiple destinations, ensure communications resiliency, and so forth. Because an SOA environment potentially is composed of multiple servers running different operating systems, the ESB functions can be distributed over several servers to form a logical Enterprise Service Bus composed of multiple instances.

Every server platform that hosts services in an SOA environment needs to take part to the Enterprise Service Bus.

The benefits of the ESB pattern are presented in Figure 8-3, which is based on a travel agency example. The ESB is the backbone of the SOA environment, which connects the business services to check flight or hotel availability for a single journey. It enables the easy addition of new services (here: Check Traveller service) and the replacement of existing ones by new ones (here: Old Flight Availability Service is replaced by New Flight Availability Service).

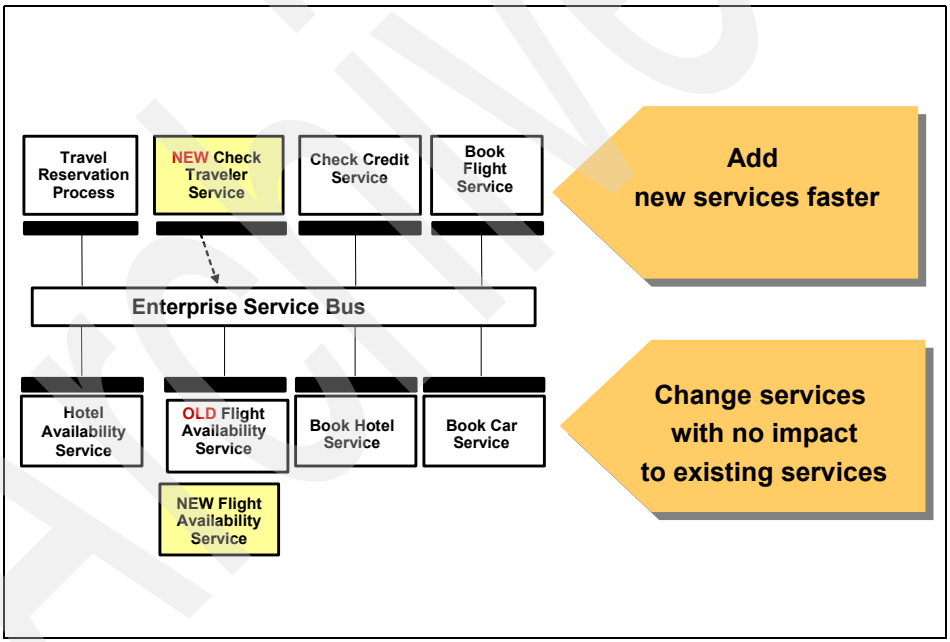


Figure 8-3 The business value of the Enterprise Service Bus

Several IBM offerings embody the ESB pattern on the System z platform:

- WebSphere Enterprise Service Bus for z/OS: An extension of the Service Integration Bus (SIBus) capabilities that are available in WebSphere

Application Server. It delivers an Enterprise Service Bus that facilitates SOA for Web services.

- WebSphere Message Broker for z/OS: Delivers an Enterprise Service Bus, facilitating SOA for both Web services and non-Web services enabled applications.

Sequence flow logic server

Processes are the coordinated chains of activities that produce a business result. They define how the business runs at an operational level. Key business processes typically cross organizational boundaries. For example, when a customer orders an item, he gets in touch with the sales division, which must be aware of stock or production status managed by the production division. They consist of:

- Human tasks such as getting a customer to sign a document, filing a document in a file cabinet, approving or refusing a request, and so forth.
- Automated tasks done via business systems with or without input from a human operator, such as opening a bank account within a bank system.

The tasks that make up a process are not always apparent. They might be hidden or embedded in the organization or culture of a company. They might be hidden in the business systems as well. It happens that when official processes are too long or impractical due to changes in the environment, users need to devise alternative bypasses.

The sequence of tasks is external to the tasks themselves and should take exceptions into account. The automation of business processes can bring a tremendous improvement in business efficiency, but it bears the risk of increased rigidities if rules cannot be adapted quickly enough to a changing environment.

We have seen that many traditional IT applications embed processes and tasks in the same container. Changing a process then requires modification to the application, which might be long and costly. In an SOA environment based on business services, a process server brings at the same time the classic benefits of automation and the flexibility of easily changeable rules. It also enables a key business requirement that has always been difficult to satisfy: process traceability.

End-to-end traceability means that you know at which step of a process a particular process occurrence is at a given time. It is like knowing where a parcel is that you have sent between its departure and arrival locations. It helps to gather overall business metrics (such as numbers of orders fulfilled the same day) and not only business division metrics.

The IBM offering for this domain on z/OS is an extension of WebSphere Application Server called IBM WebSphere Process Server. It delivers robust process automation, advanced human workflow, business rules, application

integration, and ESB capabilities on a single, integrated platform with native Web service and Java Message Service (JMS) support.

Figure 8-4 illustrates how the service bus and the Process Server extend the Application Server.

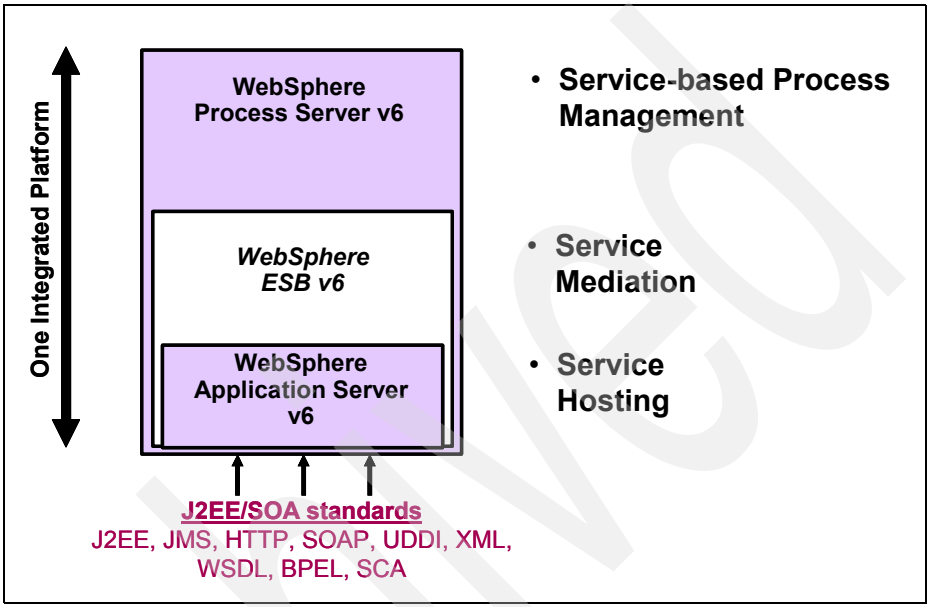


Figure 8-4 SOA in one integrated platform

8.4 SOA environments: the System z platform effect

The previous section showed that the System z platform can participate fully in an SOA environment by making its existing business services available to the enterprise. This section discusses the benefits that the System z platform brings to an SOA environment, such as how the intrinsic technical strengths of the System z platform play in the SOA world and can be linked to real business benefits.

As outlined in the first chapter of the book *Secrets of SOA*[1], Paul DiMarzio, SOA Strategist at IBM, points out:

> The platform-agnostic nature of SOA development technologies has led many pundits who watch this space to follow the path of least resistance and take up a matching platform-agnostic view of SOA deployment. Having observed several large enterprises that are reaping the benefits of SOA, as well as others that are struggling, I have concluded that deployment does indeed

[1] More information is available at http://www.secretsofsoa.com

matter—it matters a lot. So much so that it requires as much, if not more, attention than development. In fact, it has become apparent to me that larger and more complex enterprises need to take even greater care to ensure that SOA services are deployed on platforms best able to handle the job properly.

The true business value of SOA is only realized when the technology is managed and deployed with an enterprise perspective, and not in isolation. Far too often, he says, companies take a departmental “silo” approach to deployment, virtually negating the benefits of the revolutionary technology.

8.4.1 Criteria for service location

We tend to agree with the statements made by Paul DiMarzio above. In an SOA environment, a business service should not run everywhere just because it can run anywhere. Its placement should be defined to help realize the promises of SOA: better flexibility, better business knowledge, lower costs.

We mentioned earlier that in an enterprise approach, the true business services are parts of a shared application infrastructure. A shared function is more sensitive to inappropriate qualities of service:

- When it does not perform, lots of users are affected.
- When it goes down, lots of users sit idle.
- When the underlying server infrastructure needs to scale, it must be upgraded transparently to avoid creating outages running processes.
- When the underlying software stack needs an upgrade, the shared service must still be available.

The System z platform provides those qualities of service.

Some critical technical functions can get the qualities of service mentioned above (high performance, high availability, scalability) by running on multiple clone servers. For example, HTTP serving is critical but most installations use physical or logical server farms for this function.

But an HTTP server is only a technical function, not a sharable business function. Business functions are more likely to have more complex challenges to cope with. For example they must commit data with total confidence between multiple operations occurring at the same time on the same records. They should provide a rollback to a point of consistency if needed.

If a reusable business service can be defined from a current application environment such as CICS or IMS on the System z platform, it makes sense to leave it on its original platform where it can still be used for existing workloads. The System z platform has plenty of potentially reusable services to offer.

But what if a new service needs to be implemented? The following criteria must be evaluated:

- Services topology: What are the other services that this service will communicate with? Are they mostly located on a given platform or dispersed among many platforms? Although services can be called over a network, locating most related services on a single node makes sense in terms of simplicity of maintenance and operations.
- Service usage: Will the service be shared between several business functions? How many times per hour will each of these function call the service? A heavily used service should be located on a very available, scalable platform.
- Availability and resiliency: What level of availability is expected from the service? Is it supposed to run permanently? What high-availability mechanisms are in place today that the service could benefit from? What do these mechanisms protect from: power failure, network failure, server failure, operating system failure, database failure, service occurrence failure? How do they take into account server upgrades, operating system upgrades, database upgrades, upgrades to the service code (migration from one version level to a higher version level)? The System z platform provides the highest levels of availability and resiliency through mechanisms that have been presented in the previous chapters.
- Scalability: What resources will be needed to run the service in terms of processor/memory, disk space, bandwidth? What is the peak-to-low usage ratio? Is the usage predictable or very erratic? What can be the evolution paths in case the service usage outgrows the currently allocated resources? Can the service be duplicated on other similar platforms or does it need to scale on its current environment (for example, because of limitations associated with database access)? The potential growth factor of workload System z9 EC is 1 to 100 on a single server, and 32 System z servers can take part in a Parallel Sysplex cluster. This level of scalabilty can be achieved without any disruption of service.
- Management disciplines: which department will be in charge of running the service? Does it have the experience to master operations in case of unexpected events? System z platform teams are used to running critical business functions and have built procedures than are very helpful to handle problem management and change management.
- Data placement: Which data will the service manipulate? Does this data already exist? If yes, where is it? Will the data be shared between various services? Is it important that the data is up to date? Is it appropriate to offload data on a regular basis from an operational system to another system? What are the expected benefits of placing the service and its data on the same platform? The System z platform handles a high percentage of the operational data of large enterprises. It is a natural hub server for enterprise

data as presented in next chapter. It makes sense on operational terms to locate service and data on the same platform.

- Security: What level of security will the service have to provide? Should the service be accessible by identified users only? What level of end-to-end security should be available for the application if it is composed of services on different technical platforms? Is there any need to have an audit trail recording who used the service and for what purpose? Security remains a key reason why some enterprise architects continue to shy away from SOA. Security is exposed every time a service passes a communication link. With built-in security, the System z platform is attractive to reduce exposures that the dispersion of components can bring. Identity management in an SOA environment where applications are divided into pieces can be complex. The System z platform can participate in a cross-company SOA secure environment thanks to new software such as Tivoli Federated Identity Manager for z/OS systems. This software enables businesses to collaborate and manage identities and resources throughout multiple companies in a secure manner. The software secures network traffic across firewalls. Because Tivoli Federated Identity Manager covers both distributed and mainframe environments, SOA users will have secure access to all parts of their enterprise IT infrastructure.

8.4.2 Criteria for building an SOA infrastructure

The previous section dealt with the reasons to place services on the System z platform. In this section, we discuss how to locate the different elements that make an SOA infrastructure. All are available on z/OS. What are the main criteria to choose a location for the portal functions or the Enterprise Process server functions? How do you define the role of the System z platform toward the ESB?

The information services and the questions regarding data placement are discussed in the following chapter.

Where to place the portal server

The role of the portal server as an aggregation tool for presentation is a key component of the SOA infrastructure. Initially, the portal was mostly seen as a way to consolidate access to different Web applications according to user profiles and to provide a way to display information. Now with the advent of applications composed of portlets, the role of the portal within the SOA infrastructure is emphasized. For these two roles, the need for high qualities of service is strong: an enterprise cannot close the gateway to its information system.

Portlets can now be run remotely from a portal. It allows the portal infrastructure to become more distributed. It can include several portals that run on various platforms including the System z platform, and the use of the zAAP specialty

processor means that the Java workloads of the portal can be run at an attractive cost on the System z platform.

WebSphere Portal on z/OS leverages the self-configuring, self-healing, self-optimizing, and self-protecting z/OS platform to deliver workload management, scalability, near-zero downtime, and service level agreement management guaranteed results.

ESB on z/OS as the backbone for the SOA

The ESB is the key infrastructure component in the SOA architecture. It is the backbone for all communication between services. It therefore requires the highest level of availability, scalability, security, and performance. When existing core application functions on the z/OS platform are integrated using the ESB, it must provide at least the same level of Quality of Service as the backend services it accesses. This is one reason to position the ESB on z/OS. “Data proximity,” co-locating the ESB on the same platform as the backend services, provides additional levels of security, availability, scalability, and integrity. This is a second compelling reason to deploy SOA on z/OS. Cost is a last factor that will influence the decision where to deploy the ESB. Much of the ESB workload is Java workload that can be offloaded to zAAPs, which will lead to a very cost-effective configuration.

However, we must emphasize that positioning the ESB is not a question of one platform or another. Although it is possible to let the ESB on z/OS handle all requests, a logical ESB could be spread over multiple platforms. An ESB could be positioned on a distributed server for those services that do not require the high QoS of the mainframe and deal mainly with non-System z services.

z/OS as the process engine for core business processes

In the SOA environment, the services cannot fail and neither can the server that put them together. This is a strong reason for thinking of the System z platform. In most companies the core business applications run on the z/OS platform because these applications require the qualities this platform provides. Several other arguments speak in favor of the System z platform:

- Proximity: Fewer physical components and network connections result in a less complex infrastructure
- Integrity: System-managed resource recovery and transaction coordination
- Highest possible Qualities of Service
- Security: Most stringent access control and asset protection
- Availability: z(ero downtime)/OS brand promise 99.999% availability at the application and data level
- Managed workload sharing toward achievement of business goals

- Dynamic application of software changes
- Efficiency: Full utilization of system capacity with same class of service
- Fewer people to configure, monitor, and adjust workload

System z Application Assist Processors (zAAPs) execute Java workloads under z/OS at a very reasonable cost.

z/OS is the platform for core application services

New applications built on the J2EE platform require that the Mainframe Qualities of Service can be run in WebSphere Application Server for z/OS, thereby taking advantage of the cost-effectiveness of zAAP. The traditional transaction managers IMS and CICS also are very well positioned for the development of new functionality. Both platforms can participate fully in an SOA by exposing their functionality as Web services, and CICS can also act as a consumer of Web services. This is a very attractive option for the large number of customers who have deep investments in mainframe applications, need the mainframe for its qualities and facilities such as batch processing, and want to reuse these assets.

Employing and renewing existing System z platform skills

SOA brings new opportunities for employing and renewing z/OS skills. If a business service is written in COBOL or PL/1, its integration to the SOA environment and its maintenance will have to be done by professionals who understand these environments.

To bring more commonality between the classic development languages and the Java platforms, new development tools like WebSphere Developer for System z enable a common interface to develop both classic and new components.

It consists of a common workbench and an integrated set of tools that support end-to-end, model-based application development, runtime testing, and rapid deployment of on demand applications.

It offers Integrated Development Environments (IDE) with advanced, easy-to-use tools and features to help WebSphere, CICS, and IMS developers rapidly design, code, and deploy complex applications.

It helps develop creative and data-rich Web-based applications with visual layout tools to help developers write JavaServer Faces (JSF), JavaServer Pages (JSP), and HTML.

WebSphere Developer for System z includes EGL, a high-level procedural language (familiar to VisualAge® Generator, COBOL, RPG, and Informix 4GL developers) that developers who are unfamiliar with Java can use to build data-driven Web applications and business logic quickly. Developers can write

and debug their applications in EGL, and WebSphere Developer for System z generates the Java code for them. Used in conjunction with JavaServer Faces (JSF), EGL enables developers to build dynamic Web applications without having to learn the Java language. This capability opens up Java development to a new class of developers who apply their business domain expertise and use their skills as procedural developers to build dynamic Web applications.

The details of Java and J2EE are hidden from the developers, too, so they can deliver enterprise data to browsers even if they have minimal experience with Web technologies. After a developer has coded an EGL program, he generates it to create Java source (or COBOL source with the optional, new EGL for COBOL Extension feature); then EGL prepares the output to produce executable objects.

8.5 Resources

- IBM Redpaper: *The Value of the IBZM System z and z/OS in Service-Oriented Architecture*, REDP-4152
- *Secrets of SOA*, ISBN 0-9776895-7-3

9

Enterprise hub for data

This chapter discusses the benefits of using a large central repository for enterprise data.

In this chapter we:

- Introduce the issues facing data serving in large organizations.
- Describe the types of data that are present on System z platforms.
- Present different scenarios for data serving in large enterprises and how this platform can help ease the burden of making data accessible and managing and securing large amounts of data.
- Discuss how DB2 for z/OS version 8 exploits the System z platform for scalability, performance, and security. We give details of how the platform is optimized for some ISV products.
- Present the role of information as a service within the service-oriented architecture (SOA) framework and the master data management principles.

9.1 The challenges about data

In Chapter 1, “A business view” on page 1, we presented the current economic trends in which companies and organizations live now and for the foreseeable future. This environment has specific consequences for the handling of one of the precious asset of companies: data.

9.1.1 A common view of data

In the Web-based society, data must be accessible all the time from multiple channels. A bank’s customers must be able to ask about their accounts through clerks at branches, through a Web site, through a telephone call center operator, and through a mobile phone, and be able to withdraw money from an Automatic Teller Machine that might belong to a competitive bank. At the same time, batch processes might be updating the database to settle transactions between banks.

The information provided must be consistent for all channels. Having separate databases for each channel and keeping them in synch is one solution, but it can become very costly in terms of resources and management efforts, especially if the data can be modified through any channel. Finding a way to have a single view of data and making it available to all channels is an attractive proposition.

9.1.2 Managing active data

The huge databases that are needed to keep pace with the business conditions that we have described create operational challenges.

Usage scalability

As the business grows, the databases have to accommodate more records and more users accessing them. The scalability of the system platform and the database manager should enable this growth and not prevent it.

Database reorganization

Through creation and deletion of records, active data is constantly degrading the technical optimizations needed for smooth and efficient operations. Databases need to be reorganized without shutting down the many applications that rely on them.

Database back-ups

To be able to restore the database to a consistent state in case of a problem, the operations team must make copies of the databases at regular intervals. The

same stringent requirements of not impacting production usage also arise for those back-up operations.

9.1.3 Securing active and passive data

Security must be part of the design goals of systems offering to handle large amounts of data. Security comes at various levels.

Encrypting data

Sensitive data should not be stored in a readable format either in the active database or in the back-up copies on cartridges or other external media.

Preventing unauthorized access

The data stored in the databases should be accessed only with sound business reasons. The database system should restrict access to sensitive parts of the content.

9.1.4 Transforming data into information and insight

Data is useful for running the day-to-day operations of a business and more. For example, sales data can be leveraged into information about customer buying patterns, customer relationships, and business trends. Medical prescription data can help explain the pathologies that a given population is experiencing. This information then becomes a basis for action that enables better target marketing efforts or information campaigns to prevent diseases from spreading.

The large mission-critical databases become even larger because of the need to group data produced in several days or weeks of operations.

To analyze available data without affecting operations data often requires a separate environment. The regular transfer of the new data to business intelligence systems should be evaluated in detail so that it does not create a burden on the operational data store.

Entity analysis helps expain in real time who is who and who knows whom. It might help to abide with stringent regulations on anti-money laundering and passenger screening. Some of these validations can even be done after making the data “anonymous.” In that way the real data remains with its owner, and the program handles anonymous records.

9.2 Data in the z/OS context

As mentioned earlier in the description of the System z environment, the platform has several file or database managers, some of which are peculiar to z/OS. This section lists a summary.

9.2.1 File systems

This section covers the most common file systems used in z/OS.

z/OS-specific file systems

The most traditional way of storing data on z/OS is using one of many file systems that are specific to the platform, such as Virtual Storage Access Method (VSAM).

Data stored in these z/OS files can be accessed by:

- A z/OS transaction monitor such as CICS, which can be called in turn by an application server such as WebSphere
- Batch operations for non-interactive updates

UNIX file system data on z/OS

- z/OS is capable of running programs that have been built with UNIX APIs. These programs can access both the classic z/OS data and a UNIX file system data managed by z/OS; the name of this facility is zFS.

9.2.2 z/OS databases

z/OS runs several databases, both exclusive and non-exclusive to System z. This section includes summaries of the two categories.

z/OS exclusive databases

The following databases are specific to z/OS.

IBM IMS Database Manager

IBM IMS Database Manager is a hierarchical data manager that is very popular with large organizations such as financial institutions and manufacturing industries because of its capability to provide large-scale databases that are accessed by a very large number of transactions.

In his 2006 address to a U.S. IMS user group, Greg Lotko (Director, IBM DB2 for z/OS) said that usage statistics continue to rise. Clients have exceeded 100 million transactions per day and process almost US$3,000,000,000,000 per day in customer transactions. Multiple clients have gone more than 10 years

without an unplanned outage. An Asian bank has exceeded $300,000,000 in savings accounts online. IBM is investing to protect IMS customer assets and provide expansion capabilities to applications, growth in databases in both size and number, growth in transaction volumes, and enhanced abilities to create applications that integrate IMS data across the business.

IMS Database Manager data can be accessed in a variety of ways:

- IMS Transaction Manager
- CICS
- Batch jobs
- A messaging system such as WebSphere MQ
- Connectors such as IMS Connect for z/OS
- Java drivers such as IMS distributed JDBC resource adapter
- Web services through IMS SOAP Gateway
- A database federation engine such as WebSphere Classic Federation Server for z/OS

Non-IBM databases running on System z

Other software vendors have provided, and continue to enhance and maintain, databases that are specific to System z such as CA-IDMS and CA-Datacom.

Data in these environments is accessed through transaction monitors, batch, and specific connectors.

Relational DB

Several relational databases have versions that run on z/OS as well as on UNIX or Windows platforms.

IBM DB2 for z/OS

The IBM relational database management system (RDBMS) offered by System z is DB2 for z/OS. It is a member of the DB2 family of databases and exploits the strengths of that family and of the System z platform.

DB2 for z/OS data can be accessed in a variety of ways such as:

- Transactions from IMS TM or CICS
- Application servers using SQLJ or JDBC (such as WebSphere Application Server)
- Distributed Relational Database Architecture™ (DRDA®) protocol

Non-IBM middleware databases running on the System z platform

Other software vendors have the latest versions of their database products running on System z and other platforms.

We can mention Adabas from Software AG and Oracle Database 10*g*. The data that is handled by these databases can be accessed through specific connectors or standards API such as JDBC.

9.3 Roles of a z/OS environment for data serving

This section describe some roles that the z/OS environment plays in data serving.

9.3.1 Data placement

Active enterprise data can be located physically either on a distributed or a centralized environment and each option has a set of advantages and drawbacks.

Distributed placement

This solution is very efficient when:

- Data is accessed mostly through a single channel such as bank branch offices.
- The cost for the network links is high.
- The needs for data consolidation are limited.

The management of dispersed data can become difficult in terms of operations when there is a large number of locations to operate and control.

Centralized placement

Centralized placement responds better to current business needs such as multi-channel access and efficient operations.

However, the data may reside on a central location without really being seen as a global entity. In most cases, large databases are divided into large chunks or partitions that are operated more or less as individual databases accessed by a given set of servers.

Although convenient at the beginning, this approach quickly creates operational issues as server resources are not shared. System z platforms offer the ability to share access to all data between processors on a single machine as well as in the advanced Parallel Sysplex clusters.

9.3.2 Data and application integration

When the decision has been made to go for a centralized data hub on a System z platform, you need to decide where to place the applications that access the data.

The following figures illustrate three options:

- Network application serving (Figure 9-1 on page 170)
- Linux on System z application serving on System z (Figure 9-2 on page 171)
- Integrated application and database on z/OS (Figure 9-3 on page 172)

Network application serving

In this option, the application servers are running on a separate environment and they access the data on the System z platform. This environment allows for data consolidation on the System z platform and results in numerous benefits as seen above. However, there are some trade-offs:

- The connections between the application servers and the System z platforms are made with external network protocols.
- Two different teams with different sets of skills are needed to operate this environment: the distributed team and the mainframe team.

Traditionally, using System z data capabilities would have required processing power of the system. Because some software licenses are based on the total capacity of the machine, it might have increased the cost of putting data on the System z platform. This can be addressed with the latest IBM offerings such as the System z9 Integrated Information Processor (zIIP) specialty engine and enhancements to DB2 for z/OS middleware, which are designed to help customers integrate data across the enterprise, improve resource optimization, and lower the total cost of ownership (TCO) for data-serving workloads.

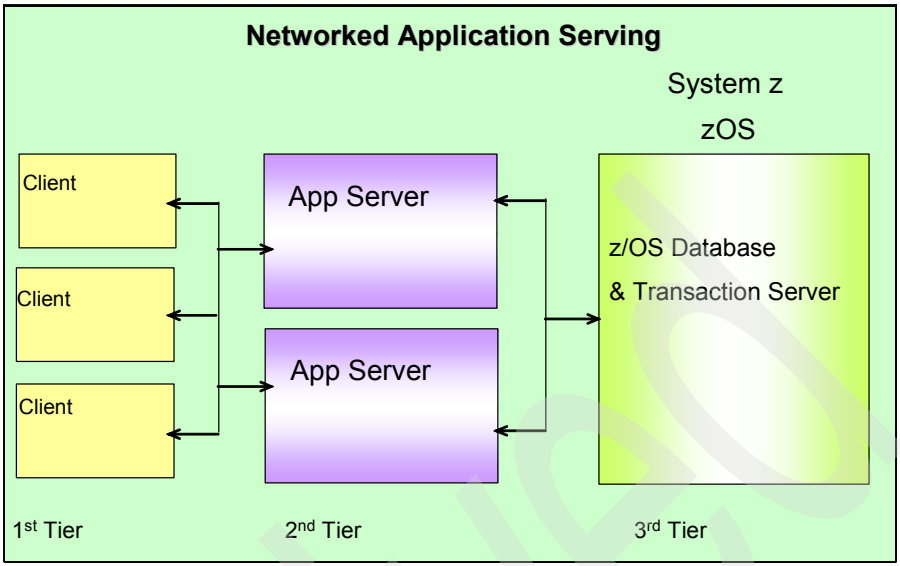


Figure 9-1 Network application serving

Linux on System z application serving on System z

With this option you locate the application servers in Linux on System z partitions on the System z platform closer to the data that resides there. Communications between partitions of the System z hardware are done through the very efficient mechanism known as Hipersockets. Hipersockets eliminate the need to utilize I/O subsystem operations and the need to traverse an external network connection to communicate between logical partitions in the same z9 EC server.

For this feature the operations team must be knowledgeable of both Linux and z/OS. It is also interesting for Independent Software Vendors (ISVs) such as SAP, who can leverage the portability of Linux applications while benefiting from the strengths of the System z platform as an enterprise hub for data.

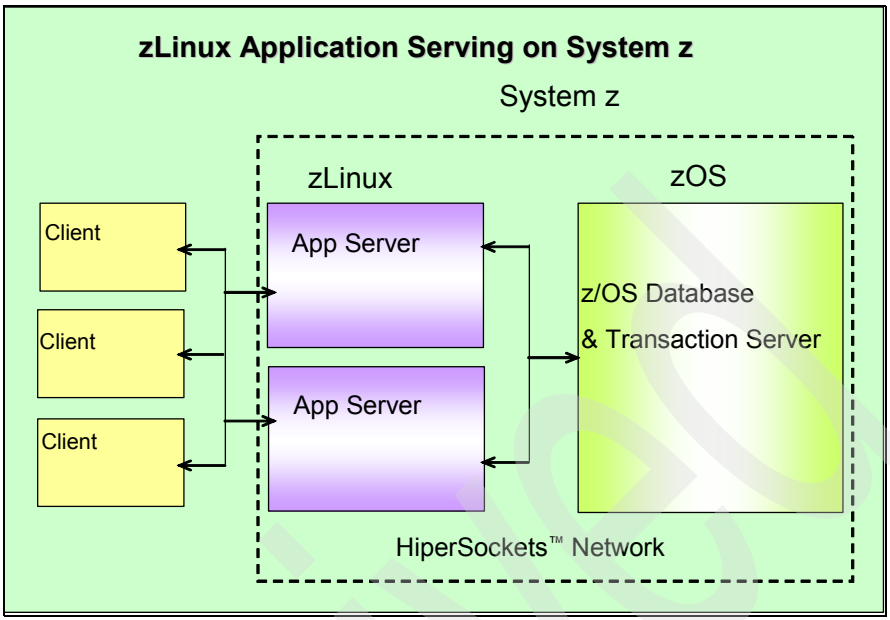


Figure 9-2 Linux on System z application serving on System z

Integrated application and database on z/OS

The applications that run directly on the System z platform under z/OS will leverage the benefits of locating applications and data in the same technical environment. Communications between the application servers and the database manager will use a very efficient cross-memory mechanism.

The operations team will have a single environment to manage.

The classic applications written in languages such as COBOL or PL/1 will run within a classic z/OS transaction manager such as CICS or IMS.

The new applications can also be written in Java and execute within CICS or IMS as well or benefit from WebSphere Application Server for z/OS, a certified J2EE application server running on the System z platform.

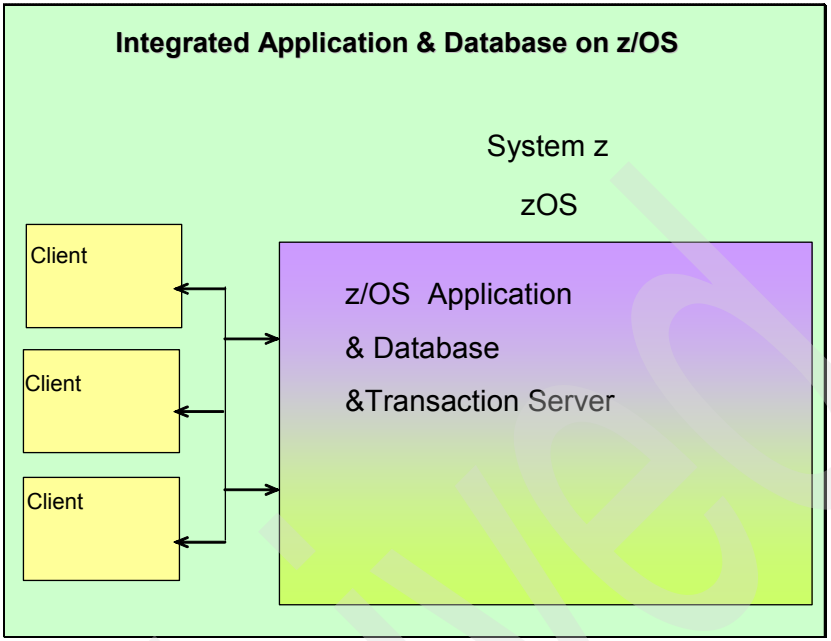


Figure 9-3 Integrated application and database on z/OS

9.3.3 Data consolidation on the System z platform

Many organizations have a lot of applications and the data that those applications manipulate is scattered in many places. Data consolidation on the System z platform without having to change the applications can bring many benefits.

Data consolidation on the System z platform helps reduce:

- The number of data copies, and hence the risk of disparate data
- The cost and complexity of backup and recovery
- The network traffic
- The amount of storage needed through centralization and efficient hardware data compression
- The database administration and management tasks
- The risk associated with distributed privacy, security, and audit policies

With data consolidation, customers leverage System z technology through:

- The use of Parallel Sysplex clustering for scalability, availability, and performance

- Data-sharing capabilities that allow for them to get a single view of data
- Centralized backup, recovery, privacy, security, and audit policies

Figure 9-4 illustrates data consolidation on the System z platform.

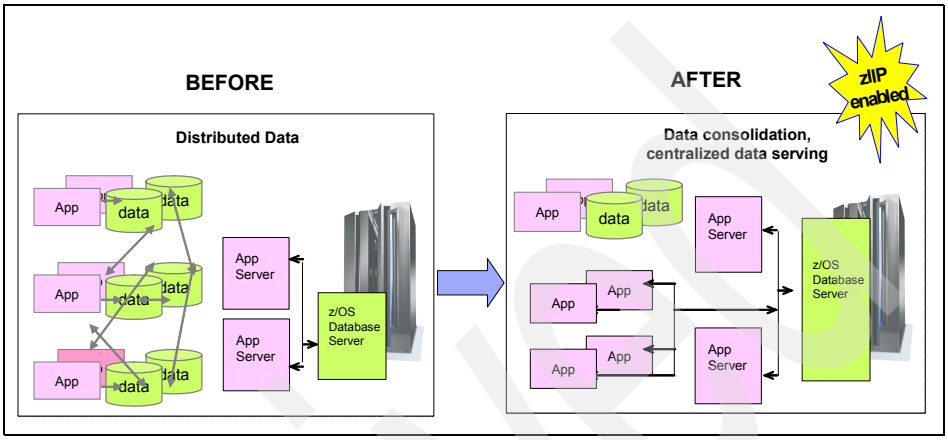


Figure 9-4 Consolidating data on z/OS

9.3.4 Data consolidation and application integration on System z

In addition to data consolidation, it is possible to integrate some applications on the System z platform with Linux on System z (Figure 9-5).

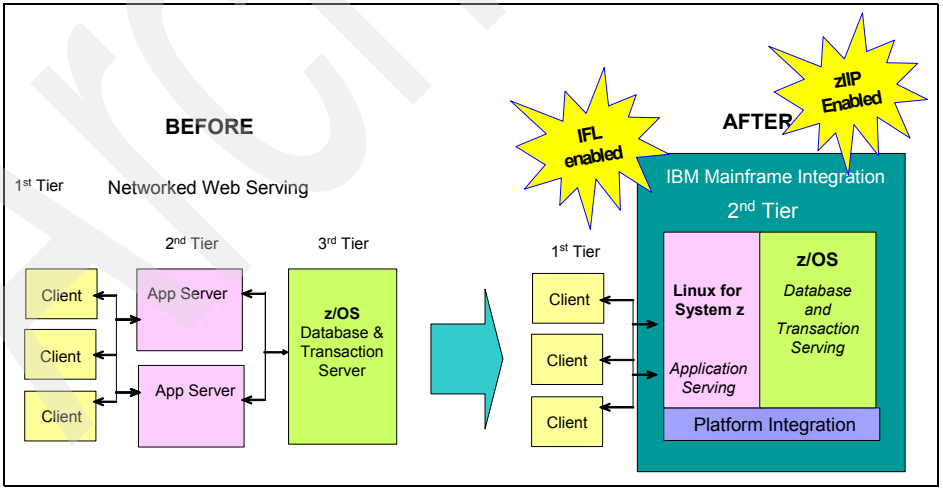


Figure 9-5 Consolidating data and application integration on the System z platform

With the integration of applications closer to the data, more benefits are available:

- There are fewer points of intrusion in a simpler network so it improves security.
- The resilience of end-to-end operations is enhanced because there are fewer points of failure. Continuous availability is easier to put in place because of the characteristics and capabilities of the System z platform.
- The portion of response time due to network latency is reduced because data and applications are located on the same platform and communicate though the very efficient Hipersockets.
- Operations team efforts are simplified because there are fewer parts to manage.
- The overall solution uses less hardware, so power and cooling needs are reduced, which improves the environmental criteria of the solution.
- The use of specialty processors such as the IBM Integrated Linux Facility (IFL) and the System z9 Integrated Information Processor (zIIP) offer those benefits at an attractive cost.

9.3.5 Data consolidation and integration of the applications on z/OS

From a technical point of view, the solution that brings the most value to the enterprise is having the data is consolidated on the z/OS environment with the applications are running there as well. Figure 9-6 shows a before-and-after illustration of this data consolidation and application integration.

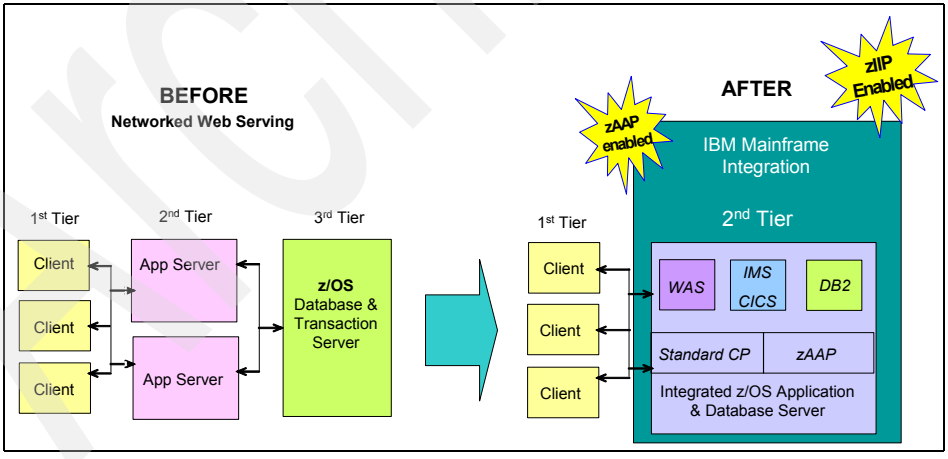


Figure 9-6 Consolidating data and integrating applications on z/OS

This situation is obvious for customers who are already running applications on the z/OS platform and need to extend them. It represents a good move in other

cases where enterprises can benefit from the portability of J2EE distributed applications to WebSphere Application Server on z/OS.

This solution increases the benefits that we already stated, and adds new ones:

- In this environment, the management of identities is more consistent, and the solution enhances auditability.
- The z/OS system is optimized for efficient use of the resources it is allowed to use.
- Transaction processing and batch work can be done at the same time on the same data: it improves availability and versatility.
- If an issue occurs, the integrated problem determination and diagnosis tools quickly help solve it.
- Automatic recovery and rollback ensure a superior level of transactional integrity.
- The Java workload created by J2EE applications can benefit from the specialty processor System z Application Assist Processor (zAAP).

9.4 The synergy between z/OS and DB2 for z/OS

IBM DB2 for z/OS is the leading relational database for the System z platform. As the following statistics show, leading global enterprises trust DB2 software on System z platforms for their business applications:

- All of the world's largest 25 banks
- Twenty-three of the top 25 retailers
- Nine of the top 10 global life and health insurance providers

The largest Online Transaction Processing (OLTP) System is the Land Registry for England and Wales. It is a 23.1 TB system and uses DB2 for z/OS, according to Kathy Auerbach, WinterCorp.[1]

> "The shipping system of the transportation services company UPS achieved a peak workload of 1.1 billion Structured Query Language (SQL) statements per hour on DB2 for z/OS".[2]

The requirements of mission-critical environments can best be achieved through deep integration of the data server with the hardware, operating system, middleware, and tools.

[1] Kathy Auerbach, Winter Corporation, Top Ten Program Press release, September, 14 2005 http://www.wintercorp.com/PressReleases/ttp2005_pressrelease_091405.htm

[2] Kathy Auerbach, Winter Corporation, Top Ten Program Press release, September, 14 2005 http://www.wintercorp.com/PressReleases/ttp2005_pressrelease_091405.htm

As a result, DB2 for z/OS delivers important benefits that are not possible from other relational database management systems on other platforms. It is this integration that enables System z servers to provide the highest levels of availability, reliability, scalability, security, and utilization capabilities as seen by the application users. That solid foundation is critical for data servers because they are at the center of enterprise applications. Any weaknesses in the underlying infrastructure are reflected all the way through the applications to users.

9.4.1 How DB2 for z/OS exploits the System z platform

DB2 for z/OS builds on the System z platform and drives some of the requirements for its evolution.

DB2 for z/OS Version 8 has been available since March 2004. It has been redesigned to leverage the 64-bit virtual addressing capabilities provided by the architecture of the System z hardware platform since 2000 and of z/OS since OS/390 Version 10. It benefits from a much larger virtual storage. The internal management tasks for very large databases have been modified to leverage this enhanced virtual storage to again improve scalabilty and availability.

Parallel Sysplex

The advanced clustering functions of the System z platform called Parallel Sysplex were presented earlier in this book. It is worth mentioning that this environment is based on the concept of “share everything” by opposition to other clustering environments that are based on the “share nothing” approach. In this latter approach some processing power is tied to a fraction of the data. In Parallel Sysplex systems, all of the DB2 data included in a DB2 group can be accessible by all of the system images participating in the cluster.

This approach is backed up by very efficient locking mechanisms that allow data accessed by several instances of an application running in different operating system images to be read or modified consistently.

DB2 data sharing support allows multiple DB2 subsystems within a sysplex to concurrently access and update shared databases. DB2 data sharing uses the coupling facility to efficiently lock data, to ensure consistency, and to buffer shared data. DB2 serializes data access across the sysplex through locking. DB2 uses coupling facility cache structures to manage the consistency of the shared data. DB2 cache structures are also used to buffer shared data within a sysplex for improved sysplex efficiency.

Accessibility

- Unicode handling

  To handle the peculiarities of the different languages of the world (accented letters, special characters, and so forth) computer users use different sets of characters named code pages. It creates a lot of difficulties to exchange data internationally.

  Unicode (http://www.unicode.org) is a set of standards that provides a consistent way to encode multilingual plain text and brings order to the “babelization” of data processing.

  DB2 for z/OS Version 8 understands Unicode, and users do not have to convert existing data. DB2 can integrate newer Unicode data with existing data and handle the translations. The synergy between DB2 and z/OS Unicode Conversion Services helps this process to be high performing.

  z/Architecture instructions exist that are designed just for Unicode conversions. There have been significant Unicode functional and performance enhancements in the System z platform starting with z/OS 1.4, z990, DB2 Version 8.

- Multiple encoding schemes

  In addition to the EBCDIC support, support for ASCII tables was added in DB2 for z/OS V5, and Unicode was added in Version 7. V8 completed the integration of multiple encoding schema support by enabling SQL access to EBCDIC, ASCII, and Unicode in the same SQL statement. The majority of the DB2 catalog tables has been converted to Unicode. Key DB2 processes such as program preparation and SQL parsing are done in Unicode.

Networking capabilities

The System z platform supports the TCP/IP V6 standard, which is the new de facto standard for interactions between nodes in a network. This capability strengthens the role of this platform as a data serving hub.

Specialty processor for data serving

The System z9 Integrated Information Processor (zIIP) is designed so that a program can have all or a portion of its enclave Service Request Block (SRB)[3] dispatched work directed to the zIIP. z/OS, acting on the direction of the program running in SRB mode, controls the distribution of the work between the general purpose processor (CP) and the zIIP. Using a zIIP can help free up capacity on the general purpose processor.

DB2 for z/OS V8 exploits the zIIP processor supported starting from z/OS V1R6.

[3] An enclave is a specific “business transaction” without address space boundaries. It is dispatchable by the operating system. It can be of system or sysplex scope.

The following types of workloads are eligible for the zIIP processor:

- Network-connected applications

  An application (running on UNIX, Linux, Intel®, Linux on System z, or z/OS) might access a DB2 for z/OS V8 database that is hosted on a System z9. Eligible work that can be directed to the zIIP is portions of those requests made from the application server to the host, through SQL calls through a DRDA over TCP/IP connection (like that with DB2 Connect).

  DB2 for z/OS V8 gives z/OS the necessary information to direct portions of the eligible work to the zIIP. Examples of workloads that might be running on the server connected through DRDA over TCP/IP to the System z9 can include business intelligence, ERP, or CRM application serving.

  Database workloads such as CICS, IMS, WebSphere for z/OS with local JDBC type 2 access, stored procedures, and Batch have become increasingly efficient and cost effective on the mainframe and are not concerned with zIIP. One key objective with the zIIP is to help bring the costs of network access to DB2 for z/OS more closely in line with the costs of running similar workloads under CICS, IMS, or Batch on the System z platform.

  Figure 9-7 illustrates the way zIIP helps reduce the workload of general processors on the System z platform for eligible workloads.

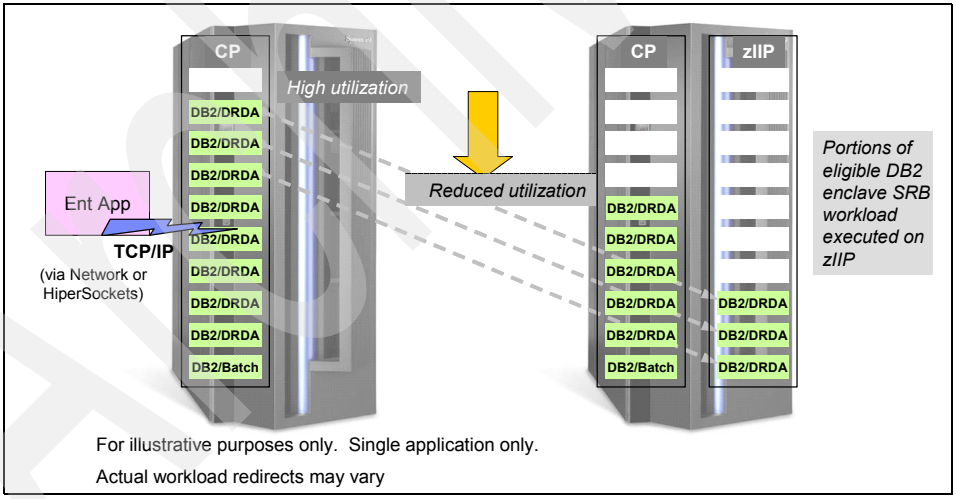


Figure 9-7 Using zIIP for enterprise applications

- Data warehousing applications

  Applications may execute queries to a DB2 for z/OS V8 database that is hosted on a System z9. Eligible work that can be directed to the zIIP is portions of requests that utilize complex star schema parallel queries. DB2 for

z/OS V8 gives z/OS the necessary information to direct portions of these queries to the zIIP. Examples of these applications can include business intelligence (BI) applications.

- Utility functions

  Some internal DB2 for z/OS V8 utility functions (Load, ReOrg and ReBuild Index) are written in SRB mode. They are performing processes related to maintenance of index structures. Those portions of those utility functions that execute in SRB mode are eligible as work that can be directed to the zIIP. DB2 for z/OS V8 gives z/OS the necessary information to direct a portion of these functions to the zIIP.

Workload management

z/OS includes policy-driven workload management functions that benefit all subsystems that are based on it, especially DB2. These functions grant workloads the right priority access to key technical resources in order to meet business goals. Workload Manager (WLM) and Intelligent Resource Director (IRD) monitor the system to adapt to changes in both workload and configuration to meet the defined goals.

Synergy with storage operations

Storage operations are a key element to support the claims of performance and efficiency of DB2 for z/OS.

DB2 for z/OS is able to share data between several systems. The storage systems help them achieve that goal.

In the context of z/OS and the IBM TotalStorage® family, five main storage capabilities reduce the need for clients to analyze their workloads and the associated data access patterns. As a result, database administrators can spend less time relocating data sets to manage data placement or bottlenecks caused by data contention.

- Multiple Allegiance (MA)

  Multiple Allegiance is the capability to support I/O requests from multiple systems, one per system, to be concurrently active against the same logical volume if they do not conflict with each other. Although Multiple Allegiance does not require software support, it is required for Parallel Access Volume.

- Parallel Access Volume (PAV)

  Normally, the operating system allows only one request at a time for each volume. With PAV, multiple unit addresses are associated with the same logical volume (aliases). These aliases are dynamically managed by WLM as a pool.

- I/O Request Priority (IORP)

  IORP is a means for associating a priority with each I/O request. Multiple applications within an operating system image may issue I/O requests concurrently. The work for each target logical volume is ordered by priority, with the number of requests matching the number of available PAV aliases.

  PAV increases parallelism for I/O requests and reduces a major cause for I/O delays. However, contention for resources may exist at some level, either in the operating system, in the channel subsystem, or in the disk subsystem. WLM I/O priority management ensures that where there is contention, the work is processed so that the work with the highest business importance is executed first. z/OS with IBM TotalStorage offers end-to-end management of I/O priorities, coordinated across even the multiple systems of a sysplex.

- Data placement

  Data placement for DB2 for z/OS partitions can be handled automatically through the autonomic functions of z/OS Data Facility Storage Management Subsystem (DFSMS). It is a set of components integrated into z/OS that enables the concept of policy-based storage management.

- FlashCopy®

  FlashCopy provides an instant, point-in-time copy of the data for application usage such as backup and recovery operations. It enables you to copy or dump data while applications are updating the data. Source and target volumes can reside on the same or different logical subsystems (LSS), also known as a Logical Control Unit (LCU). DFSMSdss utility program for data movement automatically invokes FlashCopy when you issue the COPY FULL command on a subsystem that supports FlashCopy functions.

  DB2 for z/OS V8 takes advantage of the FlashCopy functions when running utilities.

- Improved access to data

  The Modified Indirect Data Address Word facility (MIDAW) is a System z9 architecture and software exploitation designed to improve FICON channel performance. It improves the gathering and scattering of data into or from discontinuous storage locations during an I/O operation.

  Use of the MIDAW facility with FICON Express4, operating at 4 Gbps, compared to the use of classic Indirect Data Address Words (IDAWs) with FICON Express2, operating at 2 Gbps, showed an improvement in throughput of greater than 220% for all reads (270 MBps versus 84 MBps) on DB2 table scan tests with Extended Format data sets.

  These measurements are examples of what has been achieved in a laboratory environment using one FICON Express4 channel operating at 4 Gbps (CHPID type FC) on a z9 EC with z/OS V1.7 and DB2 for z/OS V8.

The performance of a specific workload may vary according to the conditions and hardware configuration of the environment. IBM laboratory tests found that DB2 gains significant performance benefits using the MIDAW facility in the following areas:

– Table scans
– Logging
– Utilities
– Using DFSMS striping for DB2 data sets

Media manager with the MIDAW facility can provide significant performance benefits when used in combination applications that use Extended Format data sets (such as DB2) or long chains of small blocks.

Figure 9-8 illustrates the performance gains.

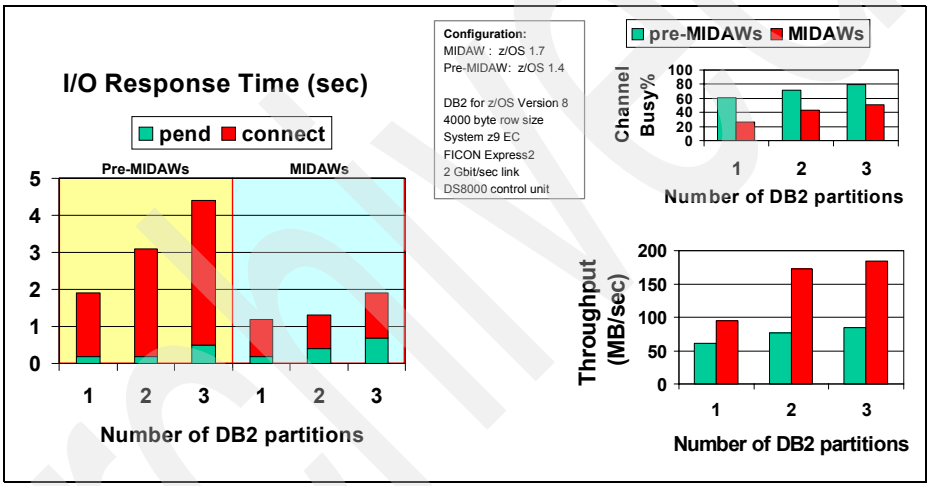


Figure 9-8 DB2 for z/OS performance improvements with System z9

- System z platform compression

  DB2 databases have been using the data compression capabilities of System z since Version 3, for table spaces (data).

Shared memory and distributed connections

Distributed connections to DB2 for z/OS will benefit from z/OS V1R7 changes that DB2 will exploit. Its distributed communication processes (the distributed address space) accesses data directly from the database manager address space, instead of moving the data. The distributed address space also exploits 64-bit addressing, as the database manager and lock manager address spaces do today with V8.

This internal change benefits new and existing workloads, where distributed communications are configured with another logical partition (LPAR) or to an application running on the System z platform.

Security synergy with Security Server for z/OS

DB2 for z/OS has strong and granular access control. It controls access to its objects by a set of privileges. Default access is none. Until access is granted, nothing can be accessed. This is called discretionary access control (DAC).

DB2 has extensive auditing features. For example, you can answer such questions as, “Who is privileged to access which objects?” and “Who has actually accessed the data?”

The catalog tables describe the DB2 objects, such as tables, views, table spaces, packages, and plans. Other catalog tables hold records of every granted privilege or authority. Every catalog record of a grant contains information such as name of the object, type of privilege, IDs that receive the privilege, ID that grants the privilege, and time of the grant.

The audit trace records changes in authorization IDs, changes to the structure of data, changes to values (updates, deletes, and inserts), access attempts by unauthorized IDs, results of GRANT and REVOKE statements, and other activities that are of interest to auditors.

You can use System z platform Security Server known as Resource Access Control Facility (RACF) or equivalent to:

- Control access to the DB2 environment
- Facilitate granting and revoking to groups of users
- Ease the implementation of multilevel security in DB2 (see details below)
- Fully control all access to data objects in DB2

DB2 defines sets of related privileges, called administrative authorities. You can effectively grant many privileges by granting one administrative authority.

Security-related events and auditing records from RACF and DB2 can be loaded into DB2 databases for analysis. The DB2 Instrumentation Facility Component can also provide accounting and performance related data. This kind of data can be loaded into a standard set of DB2 tables (definitions provided). Security and auditing specialists can query this data easily to review all security events.

Multilevel security

A multilevel security (MLS) system is a security environment. It allows the protection of data based on both traditional discretionary access controls and controls that check the sensitivity of the data itself through mandatory access

controls. It came about for centralizing sensitive data from various government agencies to provide better efficiency while preserving the confidentiality and security of data.

IBM integrated multilevel security support into z/OS starting with z/OS V1.5. Designed together with DB2 for z/OS V8, z/OS provides customers with a high assurance solution for MLS on the System z platform. This support provides row-level security labeling in DB2, and protection in z/OS, designed to meet the stringent security requirements of cross-domain access to data. This solution leverages System z platform leadership in scalability, high availability, and self-managing capabilities.

The System z platform can address requirements for highly secure data exchange. Security features in DB2 V8 and z/OS V1R5 enable customers to have a single, highly secure repository of data that has different sensitivity attributes and can be accessed by different entities and by people with different clearance levels. This secure access is managed at the row level in DB2 to provide required granularity.

Jim Porell, Chief Strategist, IBM System z software design, says:

> "Our competitive advantage with MLS is that we use the same security server for the database as we do for the operating system and some of the network communications. Therefore change management on any one of those parts is simplified because you are still preserving that centralized security manager.
>
> MLS on z/OS can help an enterprise or multiple divisions of an enterprise simplify their computing infrastructure. Compartmentalizing their data, while also consolidating the data in a large database, such as DB2 for z/OS, should reduce much of their management complexity. Hosting this database on the System z platform will provide scalability, availability, and efficient utilization levels that should meet and exceed service level needs."

You can give a group of people access to a table and limit each user to only a subset of the data based on the particular individual's MLS definition, and you can do this without creating views or placing extra predicates in the SQL. Instead, you place a security label on the data row and then associate the security label with each of the users. The data manager layer knows how to compare these security labels and can see whether a particular user is allowed to access a row. It is flexible in terms of having one set of tables and many different ways of accessing the data or providing a subset of the data.

Consequently, various entities of the same customer will be able to:

- Have faster access to merged enterprise data
- Manage multiple security classifications
- Help eliminate the need for multiple infrastructures for managing cross-domain access of data

The System z platform can provide this MLS security for applications using the latest open industry standard technologies including Enterprise JavaBeans, XML, HTML, Unicode, distributed IP networking, and PKI services.

Data encryption

System z servers have implemented leading-edge technologies such as high-performance cryptography, large-scale digital certificate support, continued excellence in Secure Sockets Layer (SSL) performance, and advanced resource access control function.

DB2 ships a number of built-in functions that enable you to encrypt and decrypt data. IBM offers an encryption tool called the IBM Data Encryption for IMS and DB2 Databases, program number 5799-GWD. This section introduces both DB2 encryption and the IBM Data Encryption tool. It also discusses recent hardware enhancements that improve encryption performance.

Data encryption has several challenges. These include making changes to your application to encrypt and decrypt the data, encryption key management, and the performance overhead of encryption.

DB2 encryption is available at the column level and at the row level.

DB2 built-in encryption features

DB2 for z/OS V8 allows encryption at the column level. Each column can have a different key. DB2 relies on the cryptographic features of System z for encryption.

Applications that need to implement DB2 encryption must apply the DB2 encrypt and decrypt built-in functions to each column to be encrypted or decrypted. Unchanged read-applications see data in encrypted form.

Some utilities support encryption, such as SQL-based DSNTIAUL, a sample unload program that can be used as an alternative to the UNLOAD utility.

IBM Data Encryption for IMS and DB2 databases

The IBM Data Encryption for IMS and DB2 databases is a single tool that enables encryption of sensitive data for IMS and DB2.

During encryption, IMS or DB2 application data is converted to database data that is unintelligible except to the person authorized by your security administrator. Sensitive data is protected at the row level for DB2 and the segment level for IMS. Encryption and decryption can be customized at these levels on the respective databases. The tool is implemented using standard DB2 and IMS exits. It exploits System z Cryptographic Hardware features, resulting in low overhead encryption/decryption.

Security and networking: SSL sessions

The System z platform provides a very efficient mechanism to support secure communications over the SSL protocol.

Security and external media storage encryption

Data administrators often think a lot about securing active data. Access is not granted to everyone and data can be encrypted as seen above.

However, the removable media storage, such as cartridges, that are used for back-up copies often contain enterprise data in readable format. If these media are stolen, enterprise data is at risk.

The System z platform provides efficient ways to secure external media storage based on hardware and software facilities.

Security certifications

The data-serving environment based on the System z platform benefits from the use of the following security certifications.

The reference information is available at:

http://www.ibm.com/security/standards/st_evaluations.shtml

- Logical partition level

  PR/SM LPAR for the IBM System z9 109 (currently z9 EC) was evaluated under the Common Criteria at evaluated assurance level 5 in Germany, level 4 worldwide. The certificate was published on March 24, 2006.

- System level

  On March 2, 2006, z/OS V1.7 with the RACF optional feature achieved EAL4+ for Controlled Access Protection Profile (CAPP) and Labeled Security Protection Profile (LSPP). EAL4+ indicates that the product was evaluated in its design stage for vulnerabilities and that the developers' testing results were independently verified. Level 4+ is a high level of independent assessed security.

- Database level

  IBM DB2 for z/OS V8 is in evaluation under the Common Criteria with a conformance claim of EAL3.

Java applications

The Java programming language is the language of choice for portable applications that can run on multiple platforms. The System z platform has been optimized to provide a very efficient Java Virtual Machine.

A new driver, the full Java DB2 Universal driver for SQLJ and JDBC offers multiplatform connectivity to DB2 databases on all platforms.

It is a single Java driver with a common code base for Linux, UNIX, Windows, and z/OS. The functions provided on DB2 for Linux, UNIX, and Windows, and DB2 for z/OS are the same, not just similar. This largely improves DB2 family compatibility. For example, it enables users to develop on Linux, UNIX, and Windows, and to deploy on z/OS without having to make any change.

XML handling

Extensible Markup Language (XML) has become the de facto data format standard to exchange information on the Internet and on corporate intranets.

IBM XML Toolkit for z/OS is designed to provide a valuable infrastructure component to assist in creating, integrating, and maintaining business-to-business (B2B) solutions. It is based on cross-platform, open source code that is designed to be compliant with industry standards.

The System z platform provides the high availability and performance needed to handle B2B and B2C-driven workloads, global application deployment, and security.

In DB2 for OS/390 and z/OS V7, if you need to create XML data from traditional relational databases (this is called XML publishing or XML composition), you had to create your own application that converts the DB2 data to the XML format, or use DB2 XML Extender.

DB2 V8 provides you with an additional option: a set of built-in functions to help with XML publishing. These DB2 built-in functions reduce your application development efforts in generating XML data from relational data with high performance, and enable the development of lightweight applications.

DB2 for z/OS V9 will provide expanded support of XML by integrating more features into the engine. This includes an XML data type, native storage of XML documents, integration of the XPath language, and catalog extensions to support definitions of XML schemas. Utilities will support creation and maintenance of XML data.

Summary

The following points are a summary of the previous discussion about the synergy between DB2 for z/OS and the System z platform.

- Availability

  The design philosophy of DB2, deep integration with System z servers and z/OS, reduces unplanned outages and enables normal maintenance without

application downtime. (Databases need not be taken down for such maintenance.)

IBM Parallel Sysplex technology is designed to prevent unplanned outages from affecting application availability.

DB2 V8 provides the ability to modify tables while the database is running, change system parameters on the fly, reorganize and load online, and conduct fast system-level point-in-time recoveries.

- Scalability

  DB2 on z/OS scales linearly when nodes are added to a Parallel Sysplex environment because the hardware coupling facility manages the cluster in a very efficient manner. In fact, when properly configured, nodes can be added to System z servers nondisruptively. In contrast, other systems have scalability limitations. Several System z hardware functions integrate tightly with DB2, which can help to improve performance and scalability. For example, System z data compression can help to save disk space and also help improve performance. Other hardware functions include sort, Unicode conversion, and encryption. With other databases, these functions are software-based and might actually degrade performance.

- Security

  The cost of security breaches can be significant for enterprises. The security and privacy capabilities of System z servers are based on many years of development efforts. In fact, over the past 20 years, DB2 has had less than five critical security patches in 2005, whereas another major vendor had as many as 89 in a single quarter during 2005.

- Total cost of ownership

  The deep integration of DB2 with server and operating system capabilities means that the database is less complex than others, simplifying operations and helping to lower administrative labor, which is the largest portion of the total cost. We have real-world customer examples in which the lower cost of mainframe management resulted in a lower overall total cost of ownership (TCO). This along with the Workload Manager (WLM) component of z/OS enables high resource utilization while maintaining response times for high-priority transactions.

9.5 DB2 for z/OS and application vendors

A great incentive for DB2 enhancements is the requirements of users and especially independent software vendors (ISVs) such as SAP AG. Major changes have been made to DB2 for z/OS in order to provide an excellent synergy for SAP data servers.

9.5.1 DB2 family of databases

DB2 for z/OS version 8 has introduced many technical enhancements that have targeted an easier port of DB2 applications from platform to platform. Some of the changes required to achieve that goal are listed as follows:

- Providing consistent table and column nams lengths.
- Enhanced SQL: More functions, better diagnostics, tighter DB2 family compatibility, pushing DB2 SQL beyond its current boundaries (for example, allowing 2 MB SQL statements).
- Longer names for tables and columns, longer SQL statements, enhancing DB2 family application portability and increasing database functionality.
- Enhanced Java and Unicode support: Improving application support and re-engineering for international business.
- Enhanced utilities: Full utility support for the extensive changes in DB2 for z/OS, plus greater DB2 family compatibility.
- The 64-bit virtual storage for all platforms to simplify main storage management and increase system availability and scalability; it helps customers who have large dynamic statement cache, many concurrently opened data sets, and long-running units of work.

Application availability has been enhanced through the ability to make changes to the database schema while running. This is called Schema evolution: Alter your table and go. There is no need to drop and redefine. It results in less system downtime and more data availability.

Having a family of databases allowing for data placement on multiple platforms is a key advantage for ISVs. They can base their developments on the IBM DB2 Universal Database™ SQL Reference for Cross Platform Development.

IBM is committed to supporting ISVs and making the System z platform an attractive platform for solution deployment.

9.5.2 ISV offerings

SAP AG

SAP is a recognized leader in providing collaborative business solutions for all types of industries and for every major market. SAP is one of the world's largest independent software suppliers overall.

The System z platform offers unmatched scalability, availability, performance, and manageability to reliably handle any information needed in a SAP solution using DB2 for z/OS as the database server.

IBM and SAP joins a proven collaboration for more than 30 years, starting with IBM as the first SAP development partner in 1972. Today the IBM and SAP alliance is a committed partnership on all levels. Global Technology and Service Partner IBM maintains full support of mySAP Business Suite on its hardware platforms and provides the largest number of SAP service consultants worldwide. Furthermore IBM is a leading and certified Partner for SAP Hosting as well as for software solutions and deployment (DB2/Information Management portfolio, Tivoli, Lotus). Focusing the knowledge and skills, experts from both companies work together in the 1993-established IBM SAP International Competence Center (ISICC). In 1999 SAP and IBM jointly announced a closer partnership at database level between SAP and DB2 development lab.

Oracle and Siebel

Oracle and Siebel applications for on demand business enable companies to draw from a consolidated source of customer information. It enables them to fully exploit multiple channels to service their customers and prospects.

To successfully run Oracle-Siebel applications for on demand business, you need a powerful, scalable database that can provide data integrity, security, and ease-of-use, such as DB2 for z/OS.

Although Oracle= has bought Siebel, customers still have the choice to run Siebel software on DB2 for z/OS and leverage their current investment with a zIIP processor.

Oracle PeopleSoft

PeopleSoft® is a major provider of applications for on demand businesses that build a collaborative network of customers, suppliers, and employees. PeopleSoft helps organizations optimize every interaction with their customers, employees, and suppliers to create more loyal, collaborative, and profitable relationships.

Many customers are deploying PeopleSoft applications in support of business-critical operations and desire to run a single global PeopleSoft instance, provide customer and supplier access to Web-based PeopleSoft applications around the clock, and support 24x7 manufacturing and distribution operations.

Those business drivers fit well with the capabilities of DB2 for z/OS.

Oracle has bought PeopleSoft, but customers still have the choice to run PeopleSoft software on DB2 for z/OS and leverage their current investment with a zIIP processor

9.5.3 More about SAP and System z platform synergy

SAP offers a comprehensive range of solutions to empower every aspect of business operations, all built on the SAP technology platform NetWeaver, such as mySAP ERP (Enterprise Resource Planning), mySAP CRM (Customer Relationship Management), mySAP PLM (Product Lifecycle Management), mySAP SCM (Supply Chain Management), and BI (Business Intelligence).

For many enterprises SAP solutions are business-critical systems with high requirements regarding data serving, availability and flexibility, but cost-effective operation. The combination of SAP solutions and the System z platform addresses these requirements exceedingly, demonstrated by means of some core value propositions.

SAP solutions are typically based on a multi-tier architecture with a presentation layer, application layer, and database layer.

Figure 9-9 illustrates the options of implementing SAP application server and database server on the System z platform.

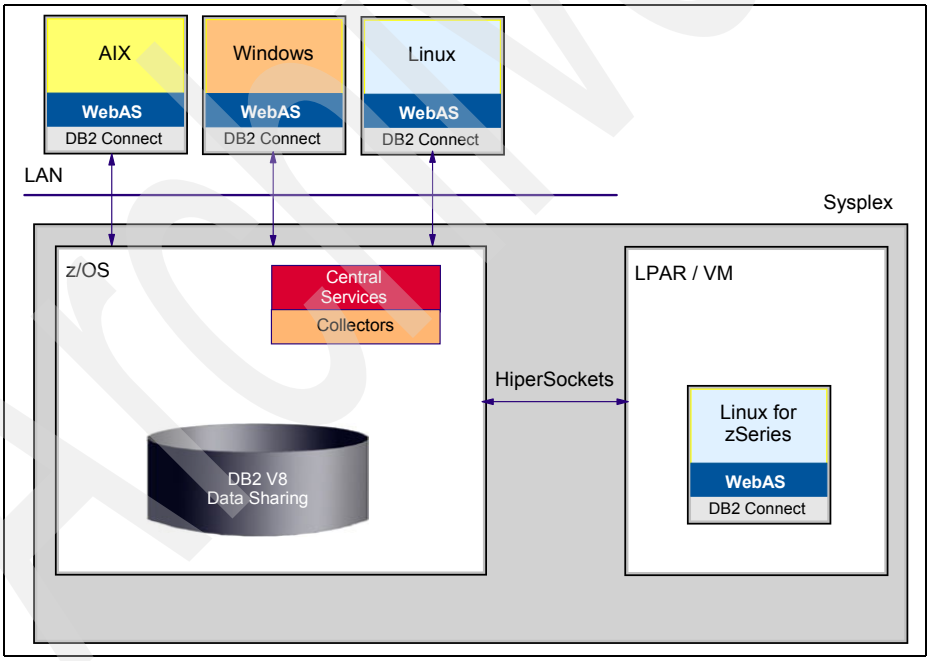


Figure 9-9 SAP Web Application Server (WebAS) and System z platform option

Scalability and availability: The database server in a SAP system landscape is unique, unlike multiple application servers, and is limited by the capacity of a

single physical server, putting significant restrictions on availability and growth. System z servers with the 64-bit capable DB2 for z/OS offer extreme scalability (enables very large SAP databases and single view of enterprise data) and ultimate availability with running a single SAP database server across up to 32 mainframe servers in a Parallel Sysplex Data Sharing mode.

On the SAP application side the most important component and single point of failure is the SAP Central Services function (messaging and enqueue processes). Placing that on the System z platform, the Parallel Sysplex technology along with use of Tivoli System Automation provides for rapid and seamless failover, leaving users unaware that any outage has taken place.

Consolidation and integration: System z is an ideal platform for consolidating various business-critical SAP solutions. Through its sophisticated virtualization and workload balancing technologies, a single server can run numerous large SAP databases and application servers side-by-side with superior performance, reducing the complexity of SAP system landscapes and thus also the associated infrastructure costs including administrative effort (TCO).

The SAP application server implementation with Linux on System z provides an especially powerful platform integration because of the highly secure internal communication with the SAP database at near-memory speed.

Additional advantages of the System z platform are also relevant if integration of SAP solutions with existing legacy corporate data and z/OS applications (for example, CICS and IMS) is required.

It provides the SAP database server on System z a great flexibility to run with SAP application servers on several other platforms, such as AIX or Windows, depending on the individual customer infrastructure and requirements.

Exploitation of zIIP: The System z9 Integrated Information Processors can be used by the workload of the SAP database server. Functions in a SAP environment being offloaded are: SQL processing for application servers using DB2 Connect, data warehousing applications with star schema parallel query processing, and some DB2 utilities.

Hardware data compression: Besides the advantage of improving performance and reducing network resources and storage (up to 60%), the highly efficient data compression on System z, exploited by DB2, also benefits the SAP database server by decreasing the extra space required by SAP Unicode implementations.

DB2 for z/OS additional capability: A distinguished advantage for the SAP database server as the most crucial part of SAP infrastructure is the outstanding strength of the System z database. Reliability and manageability of very large

SAP databases is ensured, for example with online backup functionality and very effective reorganization.

Close collaboration between SAP and IBM development led to more than 50 features of the current DB2 8 that are optimized for SAP environments.

Case study / reference:

- Baldor Electric Company, USA, builds industrial electric motors, drives, and generators. To meet its requirements for one-time delivery and cost reduction, Baldor consolidated its separate SAP applications and databases to a single IBM eServer zSeries model 990 server. The SAP database runs on DB2 for z/OS, and the SAP applications run in 24 Linux virtual machines on the same server.
- Postbank, one of the largest retail banks in Germany with more than 12 million clients, developed a new banking system based on SAP software, with requirements of highest availability and rapid recovery capabilities.

  Postbank implemented four z990 servers, distributed across two data centers in a GDPS cluster. The bank runs on z/OS, the SAP Central Services, and its very large SAP database in Data Sharing mode across the zSeries, with SAP application servers running in IBM System p servers.

9.6 Information services

Some data is used by many different applications within an enterprise:

- Party-related data such as customer name and address are used for invoicing as well as marketing messages or special bids. The same customer data may appear under several descriptions in separate applications.
- Product-related data such as description and prices are used for invoicing and stock management.
- Supplier-related data may be dispersed among various applications.

It is not uncommon that different applications use different sets of data because of segmentation of the enterprise IT environment or the merger of activities from different businesses. This situation creates technical and business issues.

From a technical point of view, the data is handled in several places. It tends to generate a lot of redundant work for creation, update, deletion, and backup, sometimes using different technologies (for platforms, databases, and so forth).

On the business side, the multiplicity of occurrences for a single party or product makes it more difficult to understand correctly who is using the company's

products or services. Several apparently unrelated customers might be part of a larger organization and be entitled to special terms and conditions that could make the company's products look more attractive.

Furthermore, not knowing precisely (or only with a lot of effort) whom your company is dealing with can create regulatory and compliance issues.

The dispersion of data between many applications and many platforms poses a great number of challenges to make it really usable. The data administrators and application designers need to:

- Access the data wherever it might be (correlating data located in different systems).
- Understand the meaning of the data, and how reliable it is.
- Cleanse the data, merge duplicate information, and correct it if necessary.
- Transform the data so it can be understood by applications, people, and processes that need it.
- Deliver the data (or make it accessible) to new destinations for new uses.

Figure 9-10 details some of the potential services around data.

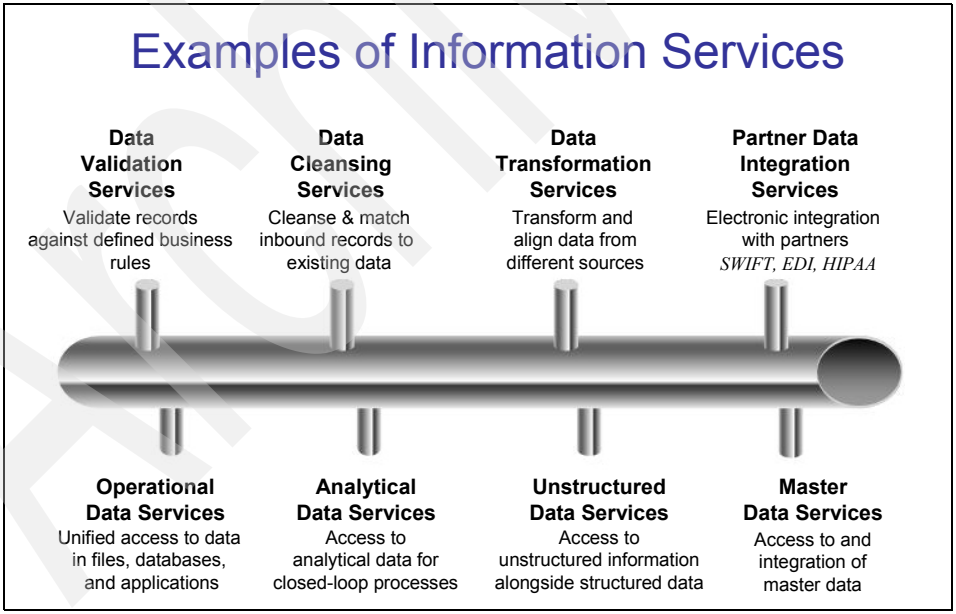


Figure 9-10 Example of information services

9.6.1 Information Integration needs

This section focuses on the main types of information integration needs.

Data federation

Data federation covers the need to provide access to diverse and distributed data as if it were in one system. One of the applications can be real-time access to System z platform data without System z platform programming. Another application could be the creation of a new virtualized data source from an existing DB2 database on the System z platform and another database on a UNIX platform.

Figure 8-11 illustrates the use of data federation to provide a single view of real-time operational data combined from three applications: claims, billing, and policy management at a large insurance carrier.

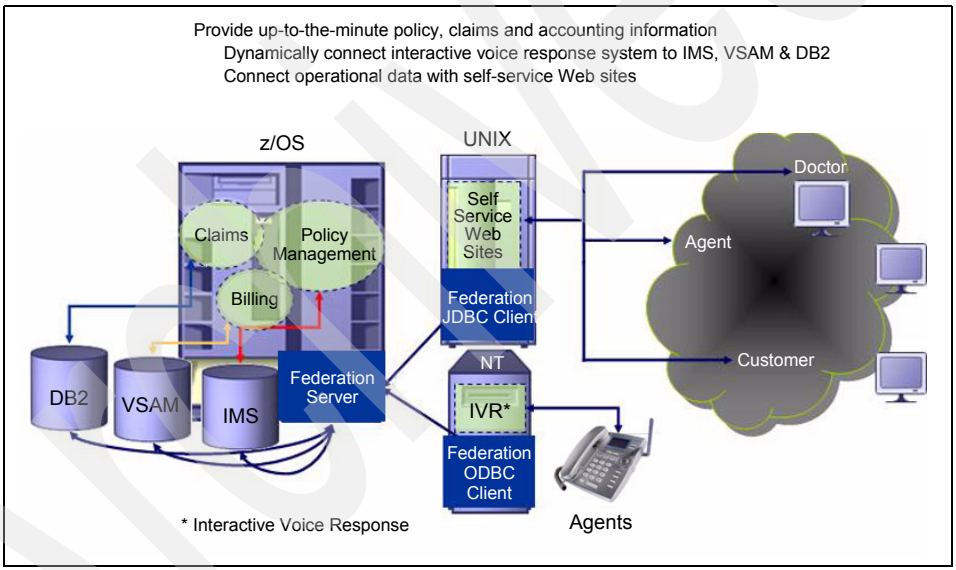


Figure 9-11 Data federation example

The original plan was to copy the data to an external database to link to an Interactive Voice System and a Self-Service System. The cost was high and data would refresh about every 36 hours because of the volume of data to be sent to the external database.

With the federation solution, made possible with WebSphere Information Integration Classic Federation, the integration work was achieved more simply.

Data replication

Data replication covers the need to synchronize data between sources.

Figure 9-12 illustrates different uses of replication for distribution to external systems, for consolidation on a System z platform, and for “cohabitation,” a situation where updates can occur on both sides.

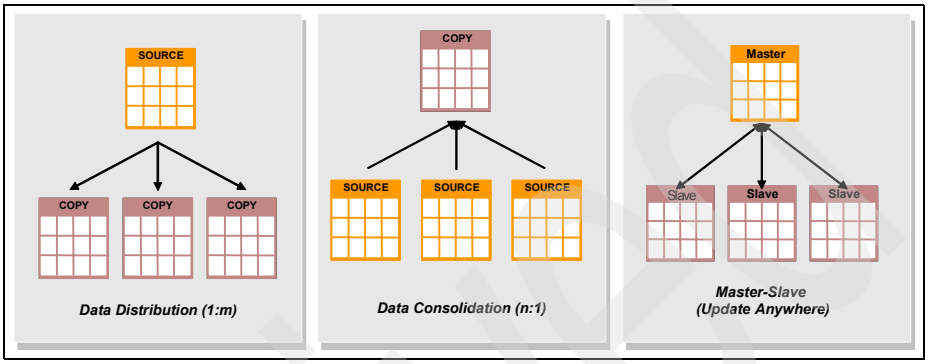


Figure 9-12 Sample replication scenarios

Data transformation

Data transformation converts source data to the different formats needed by the target systems.

Data event publishing

Data event publishing captures changes from one data set and publishes them for use in other data stores or by other applications.

9.6.2 Information integration: the System z platform offerings

With the help of IBM information integration portfolio, System z data, whether it is existing data or newly consolidated data, is easily leveraged throughout the enterprise. It enhances the capabilities of the System z platform as an “enterprise hub” for data.

Various offerings are worth mentioning:

- WebSphere Information Integrator Classic Federation for z/OS
- WebSphere Information Integrator Replication for z/OS
- WebSphere Information Integrator Event Publisher for z/OS
- WebSphere Information Integrator Classic Event Publisher:
  - For VSAM
  - For Software AG Adabas

  - For IMS
  - For CA-IDMS
- WebSphere Transformation Extender for z/OS

9.6.3 Master data management

Master data management means defining what data is commonly used by many applications in order to define a new repository that can act as the reference for this information. Master data management has many benefits.

- It enables the decoupling of master information from individual applications.
- It simplifies the ongoing integration tasks and new application development.
- It ensures consistent master information across transactional and analytical systems.
- It addresses key issues such as data quality and consistency.

Master data is critical because it defines the business context for a particular domain. Figure 9-13 shows a simplified illustration of the principles of master data management (MDM).

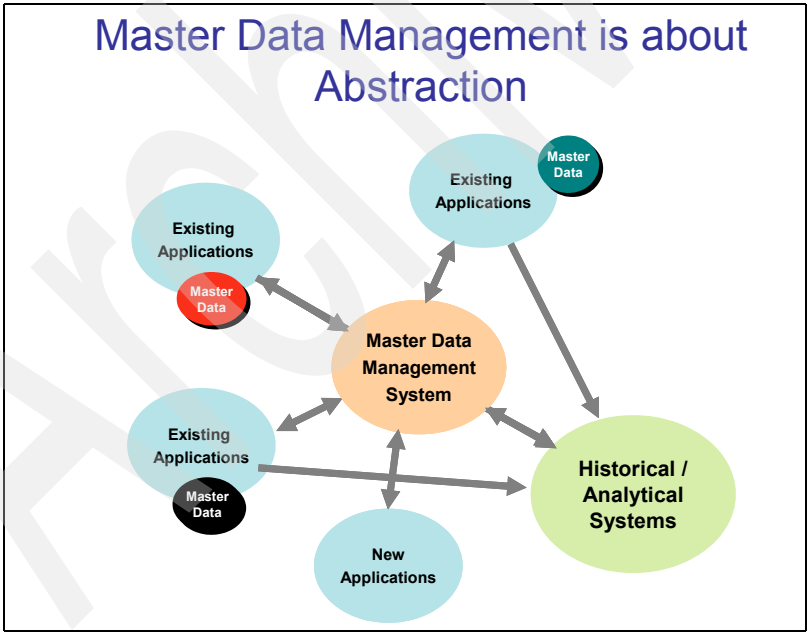


Figure 9-13 Master data management principles

IT user-related data such as user IDs and passwords might differ from application to application. This case is known as identification management. It is not handled under the MDM umbrella but can be considered very similar.

New applications will be based on the new information repositories that will result from an MDM implementation project. Existing applications cannot stop using at once the specific data and formats they have been tied to since their design. Interim solutions are needed to make those applications run while data creation and updates are delegated to the new MDM system.

The MDM process helps analyze where the master data should reside. Thanks to various tools, this data is copied or propagated to existing data stores if needed. The System z platform is uniquely positioned to take up the MDM challenge because it already hosts much reference data.

The IBM SWG offering WebSphere Customer Center leverages the z/OS platform as a data server platform.

9.7 Resources

For more information, refer to these IBM Redbooks, papers, and Web sites:

- *The Business Value of DB2 for UDB for z/OS,* SG24-6763
- *DB2 UDB for z/OS V8: Through the Looking Glass and What SAP Found There,* SG24-7088
- *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know,... and More,* SG24-6079
- *IBM TotalStorage DS8000 Series: Performance Monitoring and Tuning,* SG24-7146
- *Why data serving on the mainframe*

  `http://www.ibm.com/systems/z/feature012406/whitepaper.html`
- *SAP NetWeaver® on IBM zSeries*, see on SAP Net: SAP Service Marketplace/platform/ibm (requires password)

  `https://websmp203.sap-ag.de/platforms/ibm`
- *IBM System z9 and SAP business solutions,* ZSB01747-GBEN-00
- IBM DB2 for z/OS Web site

  `http://www-306.ibm.com/software/data/db2/zos/`
- Information about FICON, MIDAW, and FICON channel performance:

  `http://www.ibm.com/systems/z/connectivity/`

10

Conclusion

This chapter discusses the financial considerations of the System z platform. It covers:

- TCO (total cost of ownership)
- Total cost per user of the platform
- Some IBM charging policies for hardware and software that might help you evaluate platform costs

10.1 The new way of looking at the mainframe

The System z platform is known by its advantages: high availability, good redundancy, myriad I/O capabilities, and much more. But the platform is also accompanied by the common perception that it is too expensive.

But is this platform really that expensive for what it brings?

Comparing one System z server with one Intel machine shows that in some ways the System z environment is indeed more expensive, but is this the right way to compare the two platform solutions? Is the comparison between one box to one or many boxes showing the whole picture?

Reduction in the IT budget with the increased demand to give more results has caused the IT administrator to search for alternatives. Studies have shown (as summarized in Figure 1-2 on page 6) that the cost of people has increased dramatically over the past decade, and in the last few years, energy costs have been moving in the same direction. All of these trends, along with many others, indicate that the indirect cost of every IT environment is growing, and, as we know, IT professionals are looking at what they perceive to be less costly alternatives.

Experience shows that upgrading or installing a fix in a data center with hundreds or more blades or rack mount servers, most of them in a production environment, can take weeks and even longer. On the other hand, installing or even upgrading an entire System z operating system can take only seconds or a few minutes. Efficiency, achieved through automation and IT architecture, can lower administrative costs, which are the dominant component of TCO.

Calculating TCO or return on investment (ROI) of one application running on one or more Intel or UNIX servers compared to running the same application on System z will most likely show a result that is to the disadvantage of System z. If we want an accurate comparison, we should really compare all business-critical applications in the enterprise against the same number of applications running on a server farm. This kind of comparison might be difficult to make, and many people do not make it because they might not have all of the right numbers or cost information.

An article from *CIO Magazine* quoted one IT executive as saying, “ROI has more credibility when it’s stated in raw benefits, which are sometimes non-quantifiable, rather than translated into dollars. That translation is often fuzzy and tends to lose some audiences.”[1]

[1] Kalin, Sari "Return on investment, Calculating ROI", *CIO Magazine*, August 15, 2002

In this chapter we show why the System z environment can be better than others from a financial point of view. The examples we use do not quote pricing dollars, but they do try to show directions and trends.

10.2 System z functionalities

Chapter 2, “System z architecture and hardware platform” on page 15 and Chapter 3, “System z software” on page 35 provide a description of all of the specific strengths of the System z platform that contribute to the creation of infrastructures making it easy to manage, control, and monitor. All of these should be taken into consideration when determining the cost of a platform. A comparison can make sense only if it is made between platforms with similar functionalities, from both an application and a management point of view. These functionalities include:

- System management
- Security
- Data integrity
- Disaster recovery

Recent studies suggest that operational costs are one of the major costs of an installation. One way to contain or reduce this cost is to simplify the enterprise, a goal that can be achieved through automation of daily tasks or through the linearity of the architecture. Many functionalities regarding system management are built into z/OS and the hardware platform. Examples of these standard features are:

- Accumulation of accounting records to classify the users and what they consume
- Detection and recording of the system and application errors that might and might not impact the system in order to identify and diagnose potential problems, and reporting and dump generation for errors to analyze application malfunction
- Self-diagnostic and self-healing system components
- Dynamic adjustment of resources and priorities to guarantee the achievement of the desired service level
- Activation of spare hardware components (memory and processors) to guarantee continuous operations to the installation

Some of these functions are typical of and exclusive to the System z platform; other platforms might be able to acquire some similar functionalities through the purchase of additional software or hardware products that can result in additional

licensing cost and additional personnel needed to install, customize, and manage the additional elements.

Compared to other IT costs, people cost has been on the rise in the past few years. At the same time, the System z platform has proven in recent years that less personnel is required to manage and operate a growing environment from both the application and the workload points of view.

Similar considerations should be considered when you need to determine the benefits derived from how easy it is to create a disaster recovery environment.

Quantifying all of these elements is somehow installation-dependent, but there are some general rules based on the industry sector that you can use to make an initial quantification. (For example, 1.2, “Impact on IT” on page 3 shows a table with the financial loss that an installation can suffer from one hour of downtime.)

All of these elements are important when you value the functionalities that are available with each platform and, when they are not present, the cost of adding such elements to the base.

The same concept applies to the need for a disaster recovery plan with a restart window that is becoming more and more narrow and with the requirements for data integrity and a minimum data loss.

In defining the optimal target platform for an application, you need to consider the cost of maintaining application development and test environments. The System z environment is able to optimize the required resources sharing the hardware and software across multiple application environments; it is not uncommon in the System z environment to have shared servers across both the production and the development environments.

Moreover, the System z platform, through the On/Off COD function, can adjust the resources to support specific stress tests or benchmark activities, such as verifying the behavior before introducing a new and complex application. The same would be more expensive on other platforms in terms of dedicated resources and the technical personnel required to set up and manage the environment.

As described in 3.3.2, “Workload Manager (WLM)” on page 41, multiple and different workloads can coexist and share resources within the z/OS system; the Workload Manager component can manage them all according to their work priorities. In addition, with the IFL processor, it is possible to use the System z processors to consolidate distributed workloads and environments using the Linux on System z operating system. With the software virtualization provided by z/VM and hardware virtualization with PR/SM it is possible to create complex environments, often at a lower cost than when using distributed servers.

For all of these reasons, the cost analysis should consider the capability of sharing common infrastructures and understand the potential synergy with the existing workloads (for example, peaks at different times).

10.3 How to measure

There are multiple methodologies that you can use to evaluate the cost of an infrastructure. All of them have the objective to control IT costs while providing the best investment needed to cover emerging and future requirements.

However, consider other important elements when you make this evaluation, such as:

- The IT infrastructure efficiency
- The efficiency of the business directly related to your study

These elements are usually underestimated, if considered at all, during a traditional evaluation.

To perform the evaluation, one of the following methodologies can be used:

- Total cost of acquisition
- Total cost of ownership (TCO)
- Total cost per user

It is also important to understand how the budget is organized and what the trends of all the elements are in order to make a more accurate projection in time. This task is based on a preliminary cost assessment activity that is a prerequisite to any following type of analysis. During this phase, you should classify all of the different costs and assign them to the multiple platforms and multiple departments. Very often, for historical reasons, the distribution of some base and organizational costs is assigned to the main platforms (usually the System z platform) generating the feeling that these specific platforms are expensive to be managed.

Figure 10-1 shows the market trends; you can see that the software and network costs are pretty stable, the hardware cost is decreasing, and the people cost is consistently increasing.

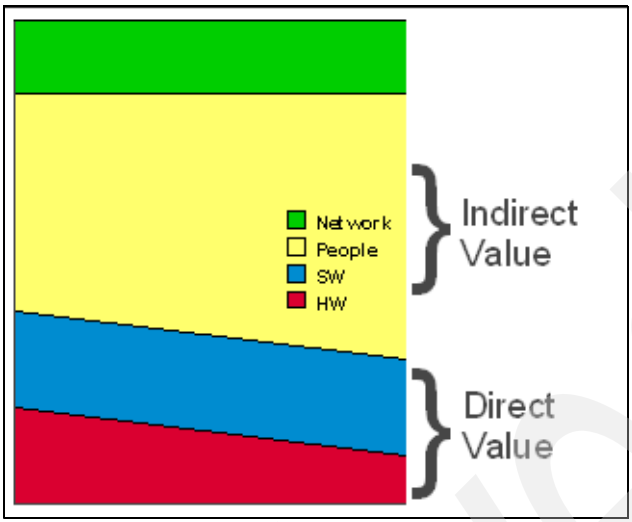


Figure 10-1 IT costs trend

As you can see, the items have also been classified as *direct* and *indirect*. One of the most noticeable things is that network costs, identified as an indirect value in the figure, are often ignored in a TCO study.

Total cost of acquisition

Total cost of acquisition (TCA) is the easiest method, although the conclusions derived by the results might be a little controversial. The method is based on the depreciation of the acquisition cost (including the financial impact in the case of leasing), for:

- Hardware
- Software licensing
- Education for new hardware and new software

The final result does not always provide the real cost that IT incurs for the acquired application or infrastructure. This method has been widely abandoned by analysts.

Total cost of ownership (TCO)

TCO started as a financial estimate to help managers classify direct and indirect costs associated with a certain investment, not necessarily only data processing. Today considerations based on the TCA methodology are incorporated in a TCO study. Because it is not based uniquely on the costs related to a limited period of

time, the TCO methodology is also called *total cost of computing*. The estimate considers:

- Direct costs
  - Acquisition cost for the hardware or software products
  - Hardware and software maintenance fees
  - Installation and customization costs
  - Recursive costs for:
    - Energy
    - Space
    - Air conditioning
  - Financial costs and residual value for the acquired products
- Indirect costs
  - Productivity of the people assigned to the management and maintenance of the infrastructure
  - People training
  - End-user training
  - System downtime (planned and unplanned)
  - Non-productivity caused by an unacceptable response time
  - Restoration from damage caused by a security issue and from an image perspective
  - Creation of development and test environments and their related maintenance
  - Creation of an effective and efficient disaster recovery environment and plan

A TCO study provides a global vision of the IT infrastructure costs that are associated with the choice and purchase of a hardware or software product.

Total cost per user

The first appearance of this methodology occurred around 2002-2004 by Arcati and Xephon analysts with the objective of providing comparable costs for the different platforms in relation to their *units of work* or *users*. The innovation brought by this methodology is the capability to provide a deeper evaluation of platforms, based not only on the financial aspects but also on the capacity of the platform itself to provide complex services to an increasing number of end users.

As discussed earlier, total cost per user comparative studies often look at the cost of running a single application, ignoring the potential that is already available in the data center. Figure 10-2 shows a total cost per user sample.

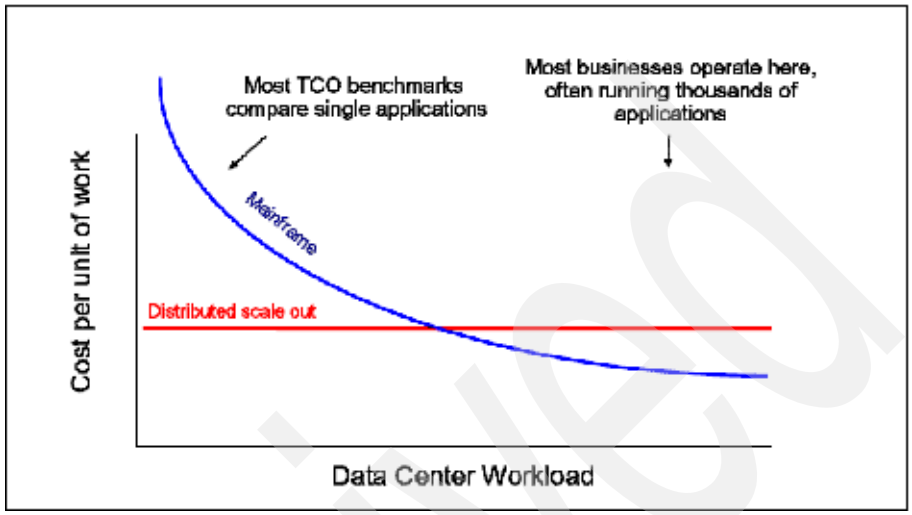


Figure 10-2 Total cost per user sample

10.3.1 References

As documented in the previous sections, many studies about infrastructure cost comparison from a financial aspect have been conducted by analysts.

Even within IBM, task forces have been working on these methodologies on the subject and manage test cases in IT centers where the need for cost analysis and optimization is present.

IBM provides a team of experts in each country that is available to provide such activity. These experts can be involved to determine the best way to evaluate infrastructure costs and to find the best solution for new workloads.

It is possible to find a detailed description of the methodologies on the IBM site. For example, information about the Scorpion methodology (used specifically to evaluate server consolidation cases comparing different platform costs) is at:

http://www.ibm.com/servers/library/pdf/scorpion.pdf

One of the first cases studied in the year 2000 contains interesting observations on the mainframe environment:

http://www.ibm.com/servers/eserver/zseries/library/whitepapers/pdf/gf225168.pdf

Recently, analysts have used the Zodiac methodology. Even this study documents the need to start with the best platform from a technical and financial point of view that is capable of handling different applications types and different products. Read it at:

http://www.gse.org/Portals/2/docs/Belgium/S03_Methodologies_for_Cross_Platform_Server_Consolidation_and_IT_Optimisation.pdf

The Linux on System z introduction and use of z/VM as a software product to expand virtualization capability on System z have been able to grow and diversify the number of solutions that can now see System z as a target platform instead of using a distributed solution. You can find additional documentation about these experiences at:

http://www.vm.ibm.com/

Looking beyond the IBM references, independent analysts offer interesting information as a result of extensive evaluation. Examples include:

- "The Dinosaur Myth - an Update," a study from Arcati at: http://www.arcati.com/papers.html
- "Mainframe Role TCO," a study from Robert Frances Group at: ftp://ftp.software.ibm.com/software/zseries/library/WhitePaper/francesroletco.pdf.

10.3.2 Sample scenarios

Every TCO study is specific to a particular installation because it depends on the initial requirements to perform the study and the specific context of each installation. This section lists two examples of TCO studies conducted for two installations that you can use as guidelines to understand the potential costs, both direct and indirect, and how to evaluate the specific functionalities of the different platforms.

New application deployment on z/OS

The deployment of a new application running on WebSphere Application Server was the originating motif for this study in order to identify the platform that had the best ratio between performance and availability point of view versus costs. The study has been requested by a financial institution where security and anti-intrusion play major roles.

The infrastructure used prior to the study was based on multiple platforms although the application core and data reside on z/OS running across a Parallel Sysplex configuration to achieve a continuous, 24x7 operation.

The study to evaluate the final platform was divided into three phases:

- Evaluation of the performance and functional requirements
- Definition of the hardware and software infrastructure requirements to achieve the objective defined in the previous phase
- Evaluation of the costs related to the potential configurations

Multiple elements have been used to provide an effective comparison of the multiple platforms, as described schematically in Figure 10-3. The study considered the efforts required, in terms of people and additional software acquisition, to obtain the functionalities on other platforms that were available on the System z platform.

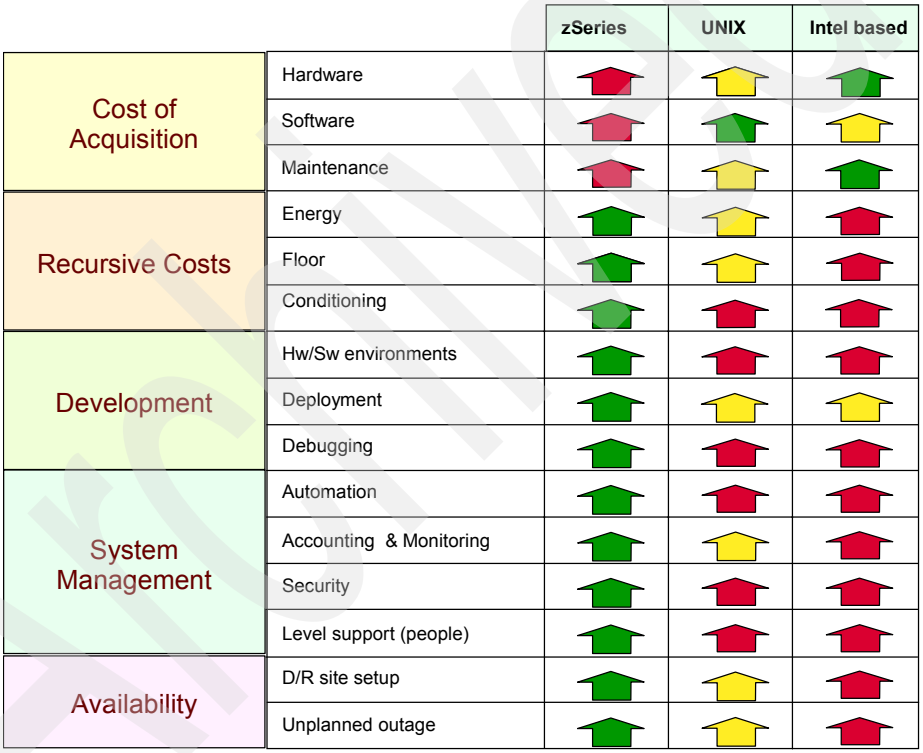


Figure 10-3 Cost evaluation: case 1

Legend: green: minimal cost increase; yellow: moderate cost increase; red: considerable cost increase

Figure 10-3 on page 208 reports the major elements that were considered during the study. Different colors of arrows are used to represent the critical costs for a specific platform. For example, you can see that the system management cost of

the distributed platform related to costs for additional software acquisition and additional support personnel is greater on the distributed platform solution.

The final evaluation of the study was in favor of the System z platform based on a four-year plan even though the initial cost of acquisition was greater for the System z platform, as shown in Figure 10-4.

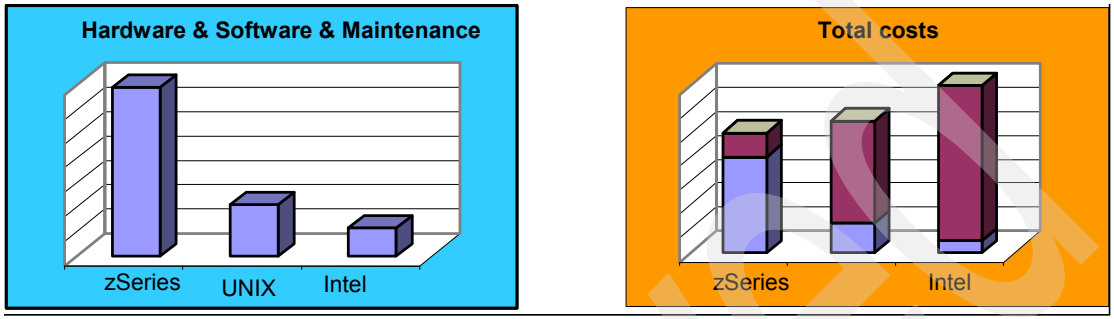


Figure 10-4 Cost comparison

For this study, the IT architect considered the zAAP processors and the hardware cryptographic functions that are available on the System z platform. In addition to these functionalities, the capability to use existing resources such as the networking and the storage environments contributed to make management of this configuration very easy from an operational and management point of view.

Server consolidation in Linux on System z and z/VM

The capability to use Linux on the System z platform provides an additional option during the server consolidation evaluation studies. In fact, Linux on System z in conjunction with the virtualization feature provided by z/VM enables a natural environment for consolidation from a distributed environment. This study illustrates that the financial advantage in consolidating a Linux server farm onto the System z platform with the virtualized infrastructure generally following these steps:

1. Identification of the distributed servers to be consolidated based on:
   - Application typology
   - Workload loads with the following characteristics
     - Peaks
     - Low frequency and duration of the peaks
     - High service availability
     - Intense interaction with DBMS running on z/OS

– Identical hardware and software configurations from a customization and infrastructure point of view

2. Comparison of the specific methodologies needed to activate the required functions for each architectural solution

Multiple TCO studies have demonstrated the many advantages that you can obtain by consolidating the Linux server farm on the System z platform. As in the previous example, Figure 10-5 shows the most important elements from a cost perspective derived by the multiple studies with an indication on the importance of each of these elements in determining the final cost for each platform.

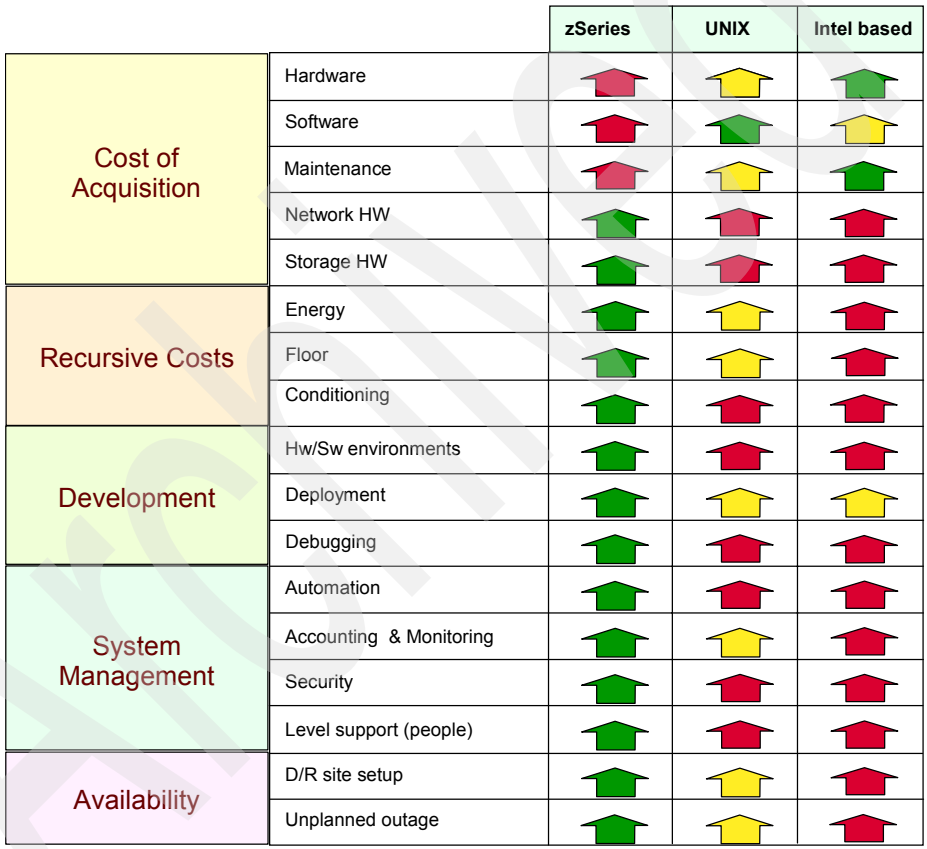


Figure 10-5 Cost evaluation: case2

Legend: green: minimal cost increase; yellow: moderate cost increase; red: considerable cost increase

Sharing networking and storage infrastructures (by use of the same libraries for the operating systems and the common middleware) provides a good economy

of scale. These functions are uniquely provided by the virtualization function of the z platform, and they enable the planning for the growth of this environment with minimal management and operation and without additional hardware cost.

10.4 Hardware and software costs on the System z

Hardware and software costs are the two elements that usually correctly considered in the evaluation and comparison between platforms. Analysts agree in recognizing the decreasing cost of the hardware.

The IBM policy is based on the recognition, through the concept of “upgrade” or MES, of the installed processing power and on the capability of offering the additional delta on top of the installed and used capacity: that is, the costs are in fact related only to the processing power and on the new functions in addition to existing functions.

In addition, toward the end of the 1990s, specialized processors were introduced to handle specific workloads such as Linux on System z or consolidated from distributed environments, or Java applications running on z/OS. These specialized processors can be used for specific workloads without being counted for software licensing.

The following sections illustrate the possibilities offered by the System z platform in regard to computing cost. That cost is influenced by new charges for the most significative platform products and by the introduction of the new specialized processor types that do not affect the software licensing.

10.4.1 MSUs versus MIPS and WLC

With the Mainframe Charter in 2003 and the Systems Agenda announcement in 2005, IBM defined a new set of rules and commercial policies regarding software licensing on System z. These rules still take into account the licensing and the monthly cost (MLC) payment methods for some of the software products such as z/OS, CICS, DB2, WebSphere MQ, and others and the one-time charge (OTC) with yearly maintenance for products such as WebSphere Application Server, DB2 utilities, and so forth.

More details about these strategies can be found at:

http://www.ibm.com/servers/eserver/zseries/swprice/

10.4.2 Software commercial strategies

To better appreciate the value of these commercial announcements, it is important to remember that System z computing power is conventionally measured in MIPS (millions-of-instructions per second), and the software licensing is based on a measurement unit called MSU (millions of service units).

The ratio between MSUs and MIPS might not be constant as MIPS are increasing or decreasing. In fact, starting in 2003, these ratios progressively decreased the cost of MIPS.

Figure 10-6 on page 212 shows these variations for the following computer hardware series:

- IBM 2064 z900 (announced in 2000)
- IBM 2084 z990 (announced in 2003)
- IBM 2094 z9 (announced in 2005)

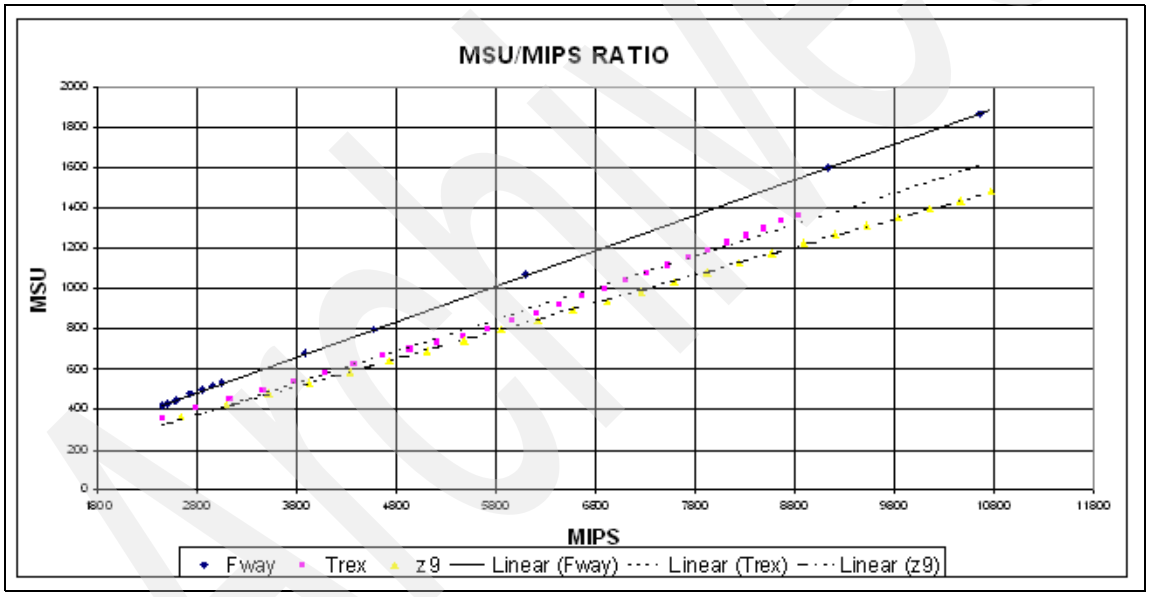


Figure 10-6 MIPS versus MSU

The curves have been stylized for reading simplicity. The slope variation represents the decreased percentage of the MSU/MIPS ratio: for example, Figure 10-6 shows that the more recent computer series provide more MIPS with lower MSU, indicating that at the same computing power the software licensing costs are significantly lower.

IBM also defined a decrease in MSU cost with an increase of computing power for the total running system; this applies to a single system (vertical growth) and to systems clustered in a Parallel Sysplex configuration (horizontal growth).

Figure 10-7 on page 213 shows an example of this methodology applied to the z/OS software licensing: An increase of computing power over 1000 MIPS generates an incremental savings for the operating system licensing cost.

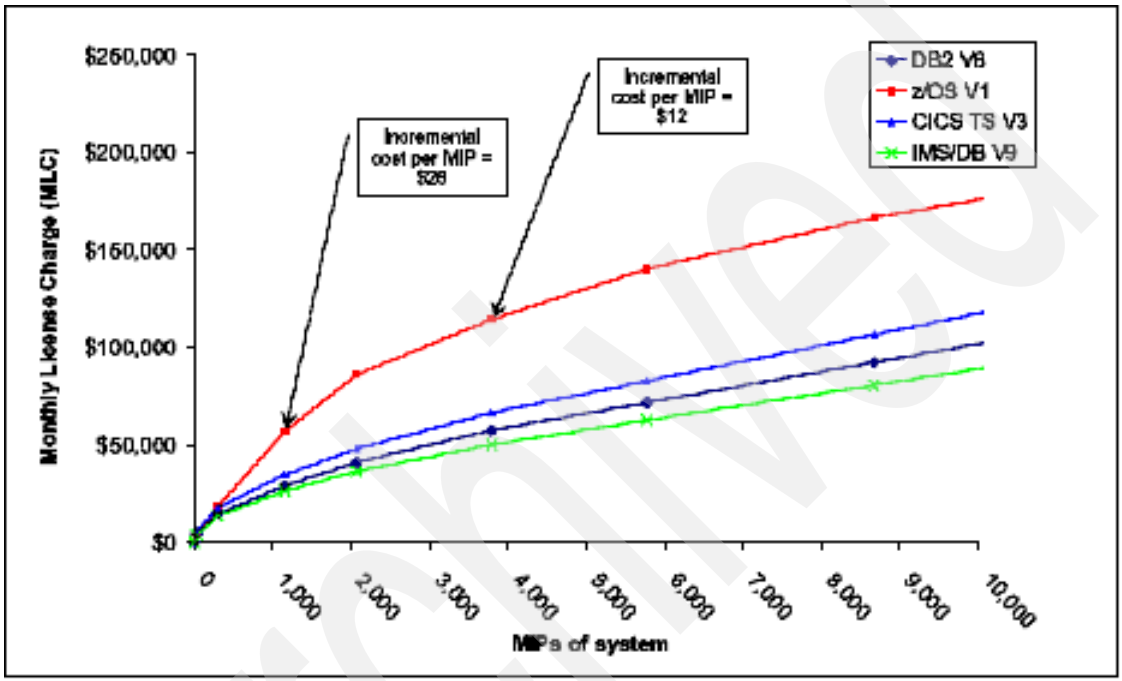


Figure 10-7 MLC sample

The year 2003 introduced a new pricing mechanism called Workload License Charge (WLC) that allows, through its Sub Capacity method, charging an installation based on the *used* computing power instead of the installed computing power. In addition, for the LPARs where applications are only using the more advanced languages, such as Java, and the more advanced infrastructures, such as WebSphere Applications Server and DB2, you can apply a different pricing model called NALC (New Application License Charge). This pricing model was designed specifically to support modern application environments such as SAP, Siebel, J2EE, and similar environments.

The Sub-Capacity Reporting Tool (SCRT) is available to help the installation define an adjustable Sub-Capacity structure. When the Sub-Capacity is in place

and working, the tool also helps define the licensing costs and checks that the defined capacity has been honored. Find out more at:

http://www.ibm.com/servers/eserver/zseries/swprice/scrt/

You can also use a PR/SM function, *soft capping*, which available on System z to keep the average 4-hour peak within the desired limit.

Another method to contain software cost is provided through the introduction of the specialized processors described in 2.3.1, “The processing unit” on page 21.

A

Additional material

This IBM Redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

`ftp://www.redbooks.ibm.com/redbooks/SG247333`

Alternatively, you can go to the IBM Redbooks Web site at:

`ibm.com/redbooks`

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG24-7333.

Using the Web material

The additional Web material that accompanies this redbook includes the following file:

File name	*Description*
SG247333.zip	Mettle Test exe file

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space: 39 MB
Operating system: Windows

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder. You should obtain the WSzOSACV2LE.exe file. Double-click the file to start the execution.

Glossary

address space. A range of virtual storage pages identified by a number (ASID) and a collection of segment and page tables that map the virtual pages to real pages of the computer's memory.

Advanced Program-to-Program Communication (APPC). 1) The general facility characterizing the LU6.2 architecture and its implementation in different Systems Network Architecture (SNA) products. (2) Sometimes used to refer to an LU6.2 product feature in particular, such as an APPC application programming interface (API)

American National Standards Institute (ANSI). An organization consisting of producers, consumers, and general interest groups that establishes the procedures by which accredited organizations create and maintain voluntary industry standards in the United States.

application program interface (API). A functional interface supplied by the operating system or by a separately orderable licensed program that allows an application program written in a high-level language to use specific data or functions of the operating system or licensed program.

application-owning region (AOR). A CICS region in an MRO environment that "owns" the CICS applications, and invokes them on behalf of remotely attached terminal (or Web) users.

application-owning region (AOR). A CICS region in an MRO environment that "owns" the CICS applications, and invokes them on behalf of remotely attached terminal (or Web) users.

authorization ID. A string that can be verified for connection to DB2 and to which a set of privileges is allowed. It can represent an individual, an organizational group, or a function, but DB2 does not determine this representation.

Capacity on Demand. The flexibility to rapidly increase or decrease computing capability as requirements change

catalog table. Any table in the DB2 catalog.

catalog. In DB2, a collection of tables that contain indexes.

central processor (CP). The part of the computer that contains the sequencing and processing facilities for instruction execution, initial program load, and other machine operations.

channel subsystem. A collection of subchannels that directs the flow of information between I/O devices and main storage, relieves the processor of communication tasks, and performs path management functions.

channel. (1) A processor system element that controls on channel path, whose mode of operation depends on the type of hardware to which it is attached. In a Channel Subsystem, each channel controls an I/O interface between the channel control element and the logically attached control units. (2) In the ESA/390 and z/Architecture, the part of a Channel Subsystem that manages a single I/O interface between a Channel Subsystem and a set of controllers (control units).

channel-attached. (1) Pertaining to the attachment of devices directly by data channels (I/O channels)to a computer. (2) Pertaining to devices attached to a controlling unit by cables rather than by telecommunication lines.

CICS. An IBM licensed program that provides online transaction-processing services and management for business applications.

client. A networked computer in which the integrated development environment (IDE) is connected to a repository on a team server. See *requester*.

CMOS. A system that controls managed systems, including the management of logical partitions and use of Capacity Upgrade on Demand. Using service applications, the HMC communicates with managed systems to detect, consolidate, and send information to IBM for analysis.

Common Connector Framework. In the Enterprise Access Builder, interface and class definitions that provide a consistent means of interacting with enterprise resources (for example, CICS and Encina® transactions) from any Java execution environment.

Common Cryptographic Architecture. A set of common cryptographic functions across multiple platforms that enable you to encipher data on one system, send it to another system, and decipher it on the destination system.

Common Gateway Interface (CGI). A means of allowing a Web server to run a program that you provide, rather than to retrieve a file. A number of popular Web servers support the CGI. For some applications (for example, displaying information from a database), you must do more than simply retrieve an HTML document from a disk and send it to the Web browser. For such applications, the Web server has to call a program to generate the HTML to be displayed. The CGI is not the only such interface, however.

control unit. A hardware unit that controls the reading, writing, or displaying of data at one or more input/output units.

Coupling Facility. A special logical partition that provides high-speed caching, list processing, and locking functions in a sysplex

CP. Central processor.

CPACF. CP Assist for Cryptographic Function.

CPCF. Central processor complex.

CPU. Central processing unit.

CU. Control unit.

CUA®. Control unit address.

DAT. Dynamic address translation.

data sharing. The ability of concurrent subsystems (such as DB2 or IMS DB) or application programs to directly access and change the same data while maintaining data integrity.

DB2 9. IBM relational database management system (RDBMS) that System z offers.

DES. Data encryption standard.

device address. In the ESA/390 architecture and the z/Architecture, the field of an ESCON or FICON (FC mode) device-level frame that selects a specific device on a control-unit image.

DFSMS. An operating environment that helps automate and centralize the management of storage.

direct access storage device (DASD). A mass storage medium on which a computer stores data.

dynamic storage reconfiguration. A PR/SM LPAR function that allows central or Expanded Storage to be added or removed from a logical partition without disrupting the system control program operating in the logical partition.

Enterprise Entity Mapping (EIM). A mechanism for associating a person or entity to the appropriate user identities in various user registries throughout an enterprise. EIM provides application programming interfaces (APIs) for creating and managing these identity mapping relationships, as well as APIs that applications use to query this information.

fiber optics. The branch of optical technology concerned with the transmission of radiant power through fibers made of transparent materials, such as glass, fused silica, and plastic.
Note: Telecommunication applications of fiber optics use optical fibers. Either a single discrete fiber or anon-spatially aligned fiber bundle can be used for each information channel. Such fibers are often called optical fibers to differentiate them from fibers used in non-communication applications.

FICON channel. A channel having a Fibre Channel channel-to-control-unit I/O interface that uses optical cables as a transmission medium. The FICON channel may operate in (1) FC mode (FICON native mode - FC-SB-2/3), (2) FCV mode (FICON conversion mode to a IBM 9032-5), or (3) FCP mode (FICON channel operating in “open mode”, which is FC-FCP).

FICON. (1) An ESA/390 and z/Architecture computer peripheral interface. The I/O interface uses ESA/390 and z/Architecture logical protocols over a FICON serial interface that configures attached units to a FICON communication fabric. (2) An FC4 adopted standard that defines an effective mechanism for the export of the SBCON command protocol via Fibre Channels.

Global Resource Serialization (GRS). A component of z/OS that serializes the use of system resources and converts hardware reserves on direct access storage device (DASD) volumes to data set enqueues.

Globally Dispersed Parallel Sysplex (GDPS). An application that integrates Parallel Sysplex technology and remote copy technology to enhance application availability and improve disaster recovery. GDPS topology is a Parallel Sysplex cluster spread across two sites, with all critical data mirrored between the sites. GDPS manages the remote copy configuration and storage subsystems; automates Parallel Sysplex operational tasks; and automates failure recovery from a single point of control.

Hardware Management Console (HMC). A system that controls managed systems, including the management of logical partitions and use of Capacity Upgrade on Demand. Using service applications, the HMC communicates with managed systems to detect, consolidate, and send information to IBM for analysis.

HiperSockets. A technology that provides high-speed TCP/IP connectivity between servers running in different logical partitions (LPARs).

IBM Encryption Facility for z/OS. A z/OS product that performs encryption and decryption of data and management of cryptographic keys.

IMS Database Manager. A database system that processes concurrent database calls and offers high performance for a variety of applications, ranging from those with moderate volumes to extremely high volumes and those with simple data structures to complex data structures.

IMS Transaction Manager. A data communication system that provides high-volume, high-performance, high-capacity, low-cost transaction processing for both IMS DB and DB2 databases.

input/output (I/O). (1) Pertaining to a device whose parts can perform an input process and an output process at the same time. (2) Pertaining to a functional unit or channel involved in an input process, output process, or both, concurrently or not, and to the data involved in such a process. (3) Pertaining to input, output, or both.

Integrated Cryptographic Service Facility (ICSF). The z/OS component that drives the System z hardware cryptographic coprocessors and that provides the IBM Common Cryptography Architecture (CCA) API for the applications and middlewares to invoke the cryptographic services.

Intelligent Resource Director (IRD). A feature of System z hardware and the Parallel Sysplex.

interface. (1) A shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics as appropriate. The concept includes the specification of the connection of two devices having different functions. (2) Hardware, software, or both, that links systems, programs, or devices.

Kerberos. A network authentication protocol that is designed to provide strong authentication for client/server applications by using secret-key cryptography.

Language Environment. An element of z/OS that provides a common runtime environment and common runtime services for C/C++, COBOL, PL/I, and Fortran applications.

Licensed Internal Code (LIC). Software provided for use on specific IBM machines and licensed to customers under the terms of IBM's Customer Agreement. Microcode can be Licensed Internal Code and licensed as such.

Lightweight Directory Access Protocol (LDAP). An open protocol that uses TCP/IP to provide access to directories that support an X.500 model and that does not incur the resource requirements of the more complex X.500 Directory Access Protocol (DAP). For example, LDAP can be used to locate people, organizations, and other resources in an Internet or intranet directory.

Linux. An open source operating system that runs on a wide range of hardware platforms and has many features that are similar to the UNIX system.

logical address. The address found in the instruction address portion of the program status word (PSW). If translation is off, the logical address is the real address. If translation is on, the logical address is the virtual address.

logical control unit (LCU). A separately addressable control unit function within a physical control unit. Usually a physical control unit that supports several LCUs. For ESCON, the maximum number of LCUs that can be in a control unit (and addressed from the same ESCON fiber link) is 16; they are addressed from x'0' to x'F'.

logical partition (LPAR). A set of functions that create a programming environment that is defined by the ESA/390 and z/Architecture. A logical partition is conceptually similar to a virtual machine environment, except that LPAR is a function of the processor and does not depend on an operating system to create the virtual machine environment.

logical processor. In LPAR mode, a central processor in a logical partition.

logically partitioned (LPAR) mode. A central processor mode, available on the configuration frame when using the PR/SM facility, that allows an operator to allocate processor hardware resources among logical partitions. Contrast with *basic mode*.

MCCU. Multisystem channel communication unit.

multilevel security (MLS). A security environmentthat allows the protection of data based on both traditional discretionary access controls, and controls that check the sensitivity of the data itself through mandatory access controls.

Open Cryptographic Service Facility (OCSF). The z/OS implementation of the Common Data Security Architecture (CDSA) API for applications running in the UNIX System Services environment.

open system. A system whose characteristics comply with standards made available throughout the industry and that therefore can be connected to other systems complying with the same standards.

parallel channel. A channel having a System/360 and System/370 channel-to-control-unit I/O interface that uses bus and tag cables as a transmission medium. Contrast with *ESCON channel*.

Parallel Sysplex. A set of z/OS systems that communicate and cooperate with each other through multisystem hardware components and software services to process customer workloads.

path group. The ESA/390 and z/Architecture term for a set of channel paths that are defined to a controller as being associated with a single S/390 image. The channel paths are in a group state and are online to the host.

path. In a channel or communication network, any route between any two nodes. For ESCON or FICON, this would be the route between the channel and the control unit/device, or sometimes from the operating system control block for the device and the device itself.

PKA. Public-key-algorithm.

Policy-based self healing. An automation policy that can be shared or cloned across the enterprise.

port. (1) An access point for data entry or exit. (2) A receptacle on a device to which a cable for another device is attached.

PR/SM. Processor Resource/Systems Manager.

processor complex. A system configuration that consists of all the machines required for operation, for example, a Processor Unit, a processor controller, a system display, a service support display, and a power and coolant distribution unit.

protocol. (1) A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication. (2) In SNA, the meanings of and the sequencing rules for requests and responses used for managing the network, transferring data, and synchronizing the states of network components. (3) A specification for the format and relative timing of information exchanged between communicating parties.

public key infrastructure (PKI). A system of digital certificates, certification authorities, and other registration authorities that verify and authenticate the validity of each party involved in an Internet transaction.

RAS. Reliability, Availability, Serviceability

Resource Access Control Facility (RACF). An IBM licensed program that provides access control by identifying users to the system; verifying users of the system; authorizing access to protected resources; logging unauthorized attempts to enter the system; and logging accesses to protected resources.

Resource Measurement Facility (RMF). A feature of z/OS that measures selected areas of system activity and presents the data collected in the format of printed reports, System Management Facility (SMF) records, or display reports.

Resource Recovery Services. The co-ordinator of transaction recovery for z/OS.

RSA. Rivest-Shamir-Adelman.

SCP. System control program.

SCSW. Subchannel status word.

Secure Sockets Layer. A security protocol that provides communication privacy. With SSL, client/server applications can communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery.

service-oriented architecture (SOA). A conceptual description of the structure of a software system in terms of its components and the services they provide, without regard for the underlying implementation of these components, services and connections between components

SMP/E. See *System Modification Program/Extended*.

subchannel. A logical function of a Channel Subsystem associated with the management of a single device.

subsystem. (1) A secondary or subordinate system, or programming support, usually capable of operating independently of or asynchronously with a controlling system.

Support Element (SE). (1) An internal control element of a processor that assists in many of the processor operational functions. (2) A hardware unit that provides communications, monitoring, and diagnostic functions to a central processor complex (CPC).

Sysplex Timer. An IBM table-top unit that synchronizes the time-of-day (TOD) clocks in as many as 16 processors or processor sides.

Sysplex. A set of systems communicating and cooperating with each other through certain multisystem hardware components and software services to process customer workloads.

System Assistance Processor (SAP). The processor that manages the storage and I/O part of the server.

System Authorization facility (SAF). An MVS interface with which programs can communicate with an external security manager, such as RACF.

System Management Facility (SMF). A z/OS facility that collects and records a variety of system and job-related information.

System Modification Program/Extended (SMP/E). An IBM licensed program used to install software and software changes on z/OS systems. In addition to providing the services of SMP, SMP/E consolidates installation data, allows more flexibility in selecting changes to be installed, provides a dialog interface, and supports dynamic allocation of data sets.

System z external time reference (ETR). The synchronization of server time-of-day (TOD) clocks to ensure consistent time-stamp data across multiple servers and operating systems. External time reference (ETR) is the MVS generic name for the IBM Sysplex Timer.

TCA. Total cost of acquisition.

TCO. Total cost of computing.

TDES. Triple Data Encryption Standard.

time-of-day (TOD) clock. A system hardware feature that is incremented once every microsecond, and provides a consistent measure of elapsed time suitable for indicating date and time. The TOD clock runs regardless of whether the processor is in a running, wait, or stopped state.

Transaction Processing Facility. Transaction Processing Facility is a specialized high availability operating system designed to provide quick response times to very high volumes of messages from large networks of terminals and workstations.

TSO. Time sharing option.

UCW. Unit control word.

Virtual memory. Addressable space that appears to be real storage. From virtual storage, instructions and data are mapped into real storage locations. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of auxiliary storage available, not by the actual number of system memory locations.

VLAN. Virtual Local Area Network.

WebSphere Application Server. Web application server software that runs on a Web server and that can be used to deploy, integrate, execute, and manage e-business applications.

WebSphere MQ. A family of IBM licensed programs that provides message queuing services.

Workload Manager (WLM). A component of z/OS that provides the ability to run multiple workloads at the same time within one z/OS image or across multiple images.

XML (Extensible Markup Lanuage XML). A standard metalanguage for defining markup languages that is based on Standard Generalized Markup Language (SGML).

z/Architecture. An IBM architecture for mainframe computers and peripherals. Processors that follow this architecture include the System z9 and eServer zSeries.

z/OS Security Server. The security environment for z/OS that incorporates various elements of security, such as user identification and authentication and access control. See **Resource Access Control Facility (RACF)**.

z/OS. An operating system for the IBM System z product line that uses 64-bit real storage.

z/VM. A System z9 operating system that acts as virtualization software. z/VM can virtualize all system resources, including processors, memory, storage devices, and communication devices. z/VM supports the concurrent operation of hundreds of operating system instances.

zAAP. IBM System z Application Assist Processor.

zIIP. IBM System z9 Integrated Information Processor.

Abbreviations and acronyms

abbreviation1	Description1
abbreviation2	Description2
IBM	International Business Machines Corporation
ITSO	International Technical Support Organization
abbreviation3	Description3
abbreviation4	Description4

Index

D

E

F

G

H

I

J

Q

R

S

Related publications

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

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 236. Note that some of the documents referenced here may be available in softcopy only.

- *DB2 UDB for z/OS V8: Through the Looking Glass and What SAP Found There*, SG24-7088
- *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More*, SG24-6079
- *IBM TotalStorage DS8000 Series: Performance Monitoring and Tuning*, SG24-7146
- *Linux on IBM zSeries and S/390: Performance Measurement and Tuning,* SG24-6926
- *The Business Value of DB2 for UDB for z/OS* , SG24-6763
- *The Value of the IBM System z and z/OS in Service-Oriented Architecture*, REDP-4152
- *z/OS Intelligent Resource Director,* SG24-5952

Other publications

These publications are also relevant as further information sources:

- *Secrets of SOA, ISBN 0-9776895-7-3*
- *z/OS V1R8.0 MVS System Management Facilities,* SA22-7630-13
- *z/OS V1R8.0 Resource Measurement Facility (RMF) User’s Guide,* SC33-7990-11
- *z/OS V1R8.0 MVS Planning Workload Management,* SA22-7602-12

Online resources

These Web sites are also relevant as further information sources:

- DB2 for z/OS

  http://www-306.ibm.com/software/data/db2/zos/

- I/O connectivity and FICON channel performance:

  http://www.ibm.com/systems/z/connectivity/

- IBM System z9 and SAP business solutions

  http://www-03.ibm.com/solutions/sap/doc/content/bin/IBM_System_z_and_SAP_solutions.pdf

- SAP Technology Paper: "SAP NetWeaver on IBM zSeries"
  See on SAP Net: SAP Service Marketplace/platform/ibm (requires password)

  https://websmp203.sap-ag.de/platforms/ibm

- Why data serving on the mainframe

  http://www.ibm.com/systems/z/feature012406/whitepaper.html

- Scorpion methodology

  http://www.ibm.com/servers/library/pdf/scorpion.pdf

- *z/Architecture Principles of Operation,* A22-7832-04

  http://publibz.boulder.ibm.com/epubs/pdf/a2278324.pdf

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

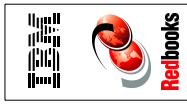
Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



IBM System z Strengths and Values

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

IBM System z Strengths and Values

This IBM Redbook describes the strengths and values of the IBM System z platform with special focus on IBM z/OS. Other System z operating systems are described briefly in relation to the values that customers can obtain from them while running on the System z hardware.

This book addresses the technical aspects of the System z environment and focuses on the business value of the System z solution, the overall role that it plays in the business enterprise, and how it offers an effective total cost of computing.

The target is the IT architect who is not familiar with the System z platform. System z architects will also be interested in this book for understanding the important role that the System z environment plays in the business solution, especially with the deployment of new workloads and to address security, availability, and scalability.

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

SG24-7333-01 ISBN 0738431028