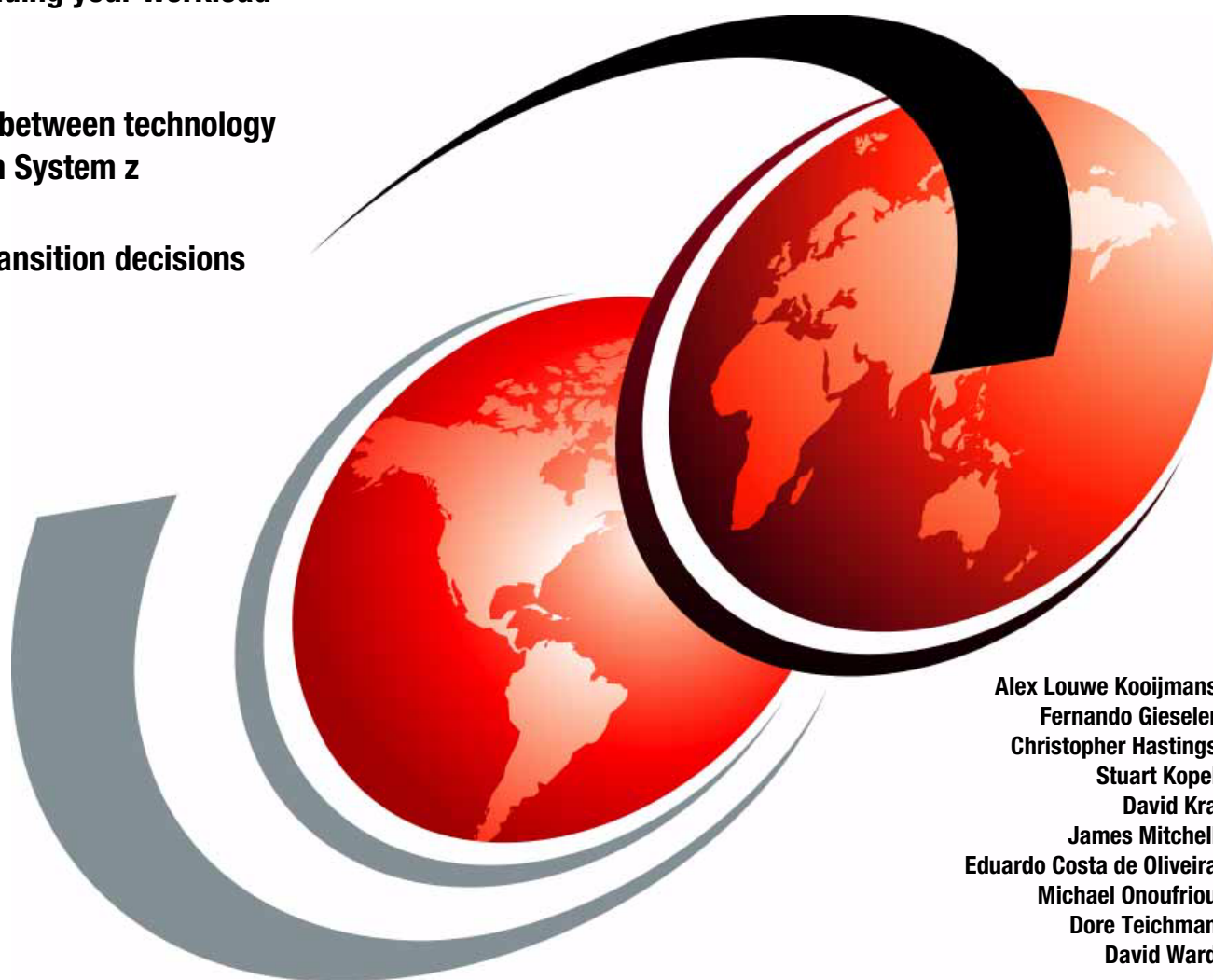


Considerations for Transitioning Highly Available Applications to System z

Understanding your workload

Choosing between technology
options on System z

Making transition decisions



Alex Louwe Kooijmans
Fernando Gieseler
Christopher Hastings
Stuart Kopel
David Kra
James Mitchell
Eduardo Costa de Oliveira
Michael Onoufriou
Dore Teichman
David Ward

Redbooks



International Technical Support Organization

**Considerations for Transitioning Highly Available
Applications to System z**

September 2011

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

First Edition (September 2011)

This edition applies to IBM System z and z196 servers running supported versions of Linux for System z or z/OS.

© Copyright International Business Machines Corporation 2011. All rights reserved.

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

Contents

Notices	ix
Trademarks	x
Preface	xi
The team who wrote this book	xi
Now you can become a published author, too!	xiii
Comments welcome	xiii
Stay connected to IBM Redbooks	xiii
Chapter 1. Management summary	1
Chapter 2. Introduction to systems availability	3
2.1 Introduction	4
2.2 Availability concepts	4
2.2.1 High availability	4
2.2.2 Continuous operation	5
2.2.3 Continuous availability	5
2.2.4 Business impact of outages	6
2.3 Evaluating availability	7
2.3.1 Key availability issues	8
2.3.2 Causes of service outage	8
2.3.3 Service Level Agreement (SLA)	9
2.3.4 Service request failure	10
2.4 Other system availability measures	11
2.5 Designing for availability	11
2.5.1 Elements of a continuously available service	12
2.5.2 Failing over to another site and using moderate quality components, or not . . .	12
2.5.3 Surviving component failure with redundancy	13
2.6 Summary	13
Chapter 3. Target architectures for availability	15
3.1 Target reference architectures	16
3.1.1 Each site	16
3.1.2 Activity levels for two-site deployments	24
3.1.3 Information replication between sites	26
3.1.4 Multisite deployment	31
3.1.5 Summary	39
3.2 Factors that influence the approach to take on the target architecture	41
3.2.1 Non-functional requirements	41
3.2.2 Fit for purpose	42
3.2.3 Cost of the target architecture	43
3.2.4 Risk	44
3.2.5 Business requirements	45
3.2.6 Mapping source architecture to target architecture	45
3.2.7 Common terminology	46
3.3 Integration with external services	46
Chapter 4. System z technology options	49
4.1 Operating systems	50

4.1.1	z/OS	50
4.1.2	z/VM	51
4.1.3	Linux on System z	51
4.1.4	Important decisions with respect to operating systems	52
4.2	Virtualization	52
4.2.1	Mainframe virtualization technologies	52
4.2.2	PR/SM and logical partitioning	53
4.2.3	z/VM	55
4.2.4	Address spaces in z/OS	59
4.2.5	Enclaves on z/OS	61
4.2.6	Important decisions with respect to virtualization	61
4.3	Resource management	61
4.4	Workload management	63
4.4.1	Workload management under z/OS	63
4.4.2	Workload management under z/VM	66
4.5	Network and communications	68
4.5.1	Support of legacy protocols	68
4.5.2	Server load distribution	69
4.5.3	System z network connectivity	69
4.5.4	z/VM virtual networking	71
4.5.5	Important decisions with respect to communications	74
4.6	Availability and scalability	75
4.6.1	Availability	75
4.6.2	Disaster recovery	75
4.6.3	What about planned outages	76
4.6.4	Parallel Sysplex	77
4.6.5	High availability solutions for data	78
4.6.6	Storage resilience considerations	80
4.6.7	Scalability	82
4.6.8	Important decisions with respect to availability and scalability	85
4.7	Security	85
4.7.1	Security in z/OS	86
4.7.2	Security in z/VM and Linux on System z	87
4.7.3	Important decisions with respect to security	95
4.8	Transaction management	96
4.8.1	Running programs in a transaction manager or not	96
4.8.2	Important decisions with respect to transaction management	97
4.9	Data management	97
4.9.1	Database management systems	97
4.9.2	Highly available file systems	105
4.9.3	Important decisions with respect to data management	107
4.10	Programming environment	107
4.10.1	Java on System z	108
4.10.2	COBOL and PL/I	108
4.10.3	C/C++	109
4.10.4	Application development tooling	111
4.10.5	Important decisions with respect to programming environment	112
4.11	Integration	112
4.11.1	Integration layers	112
4.11.2	Integration styles	113
4.12	Systems management	114
4.12.1	Availability and performance management	114
4.12.2	Capacity management and sizing	118

4.12.3	Security management	118
4.12.4	Service level management	119
4.12.5	Problem and change management	119
4.12.6	Configuration management	121
4.12.7	Release management	121
4.12.8	Storage management	122
4.12.9	Service continuity management	123
4.12.10	Incident management	123
4.12.11	Asset and financial management	124
4.13	Specialty processors	124
Chapter 5.	Operational models for high availability.	129
5.1	Introduction	130
5.1.1	Planned outages	130
5.1.2	A word on scalability and capacity	130
5.2	Single-site Parallel Sysplex	131
5.2.1	Model description	131
5.2.2	Availability characteristics	132
5.2.3	Functionality	132
5.2.4	Availability scenario	132
5.3	Single-site Parallel Sysplex with two servers and disk-level replication (basic Hyperswap)	133
5.3.1	Model description	133
5.3.2	Availability characteristics	134
5.3.3	Functionality	134
5.3.4	Availability scenario	134
5.4	Dual site with data replication	134
5.4.1	Model description	135
5.4.2	Availability characteristics	135
5.4.3	Functionality	136
5.4.4	Availability scenario	136
5.5	Dual-site Parallel Sysplex with data-level replication (active/passive)	137
5.5.1	Model description	137
5.5.2	Availability characteristics	137
5.5.3	Functionality	138
5.5.4	Availability scenario	138
5.6	Dual-site Parallel Sysplex with disk-level replication (Metro Mirror)	139
5.6.1	Model description	139
5.6.2	Availability characteristics	139
5.6.3	Functionality	139
5.6.4	Availability scenario	140
5.7	Dual-site Parallel Sysplex with disk-level replication (Global Mirror).	140
5.7.1	Model description	141
5.7.2	Availability characteristics	141
5.7.3	Functionality	141
5.7.4	Availability scenario	142
5.8	Linux on System z - Linux HA	142
5.8.1	Model description	142
5.8.2	Availability characteristics	143
5.8.3	Functionality	143
5.8.4	Availability scenario	143
5.9	Linux on z - GDPS	143
5.9.1	Model description	144

5.9.2 Availability characteristics	144
5.9.3 Functionality	145
5.9.4 Availability scenario	145
5.10 Model summary	145
Chapter 6. Workload assessment	147
6.1 Workload assessment overview	148
6.1.1 Workload tiers	148
6.1.2 Domains of Transformation.	148
6.1.3 Availability and scalability assessment	152
6.1.4 Integration assessment.	155
6.1.5 Middleware assessment	157
6.1.6 Database and file systems	159
6.1.7 Security assessment.	161
6.1.8 Systems management assessment	164
6.1.9 Capacity	164
6.2 Completing the assessment	165
6.2.1 Defining the domains of transformation matrix	165
6.2.2 Assigning complexity and risk ratings	168
6.3 Workload assessment summary	169
Chapter 7. Transition approaches	171
7.1 Introduction and overview of transitional approaches	172
7.2 Application patterns.	173
7.2.1 AP1: An application using a binary-compatible language.	173
7.2.2 AP2: An application written in a standard language supported on System z . . .	173
7.2.3 AP3: An application with a proprietary language and data store	173
7.2.4 Summary of application patterns.	174
7.3 Transitional patterns	175
7.3.1 TP1: Buy a software package or recode the application from scratch	175
7.3.2 TP2: source code conversion	175
7.3.3 TP3: Port the existing application by recompiling on System z.	176
7.3.4 TP4: Redeploy the existing application as it exists today	176
7.3.5 Summary of transitional patterns	177
7.3.6 Mapping application patterns to transitional patterns	177
7.3.7 Middleware provisioning	178
7.4 Switchover approaches.	178
7.4.1 The big bang switchover approach	179
7.4.2 Shadowing with replication	180
7.4.3 Phased switchover	182
7.5 Options for application modernization.	183
7.5.1 The case for application modernization	183
7.5.2 Aspects of application modernization	184
7.5.3 Modernization using a service oriented architecture.	184
7.6 Summary	186
Chapter 8. System z capacity planning, sizing, and TCO	187
8.1 Overview of sizing, capacity planning, and TCO considerations.	189
8.2 The importance of sizing	189
8.3 Capacity planning importance.	190
8.4 Data collection and analysis	191
8.4.1 Data collection	192
8.4.2 Workload characterization.	193
8.4.3 Processor analysis	200

8.4.4	Memory analysis	207
8.4.5	Disk analysis (I/O)	208
8.4.6	Network analysis	210
8.5	Sizing methodologies	213
8.5.1	Sizing for consolidation or migration	213
8.5.2	Modeling and sizing tools	217
8.5.3	Sizing from scratch	219
8.5.4	Sizing High Availability and Disaster Recovery	221
8.6	Capacity planning methodologies	223
8.6.1	Guidelines	224
8.6.2	Linear projections	224
8.6.3	Analytic methods	224
8.6.4	Discrete methods	224
8.6.5	Real world benchmarks versus standard benchmarks	225
8.6.6	Processor capacity planning	226
8.6.7	Memory capacity planning	227
8.6.8	I/O capacity planning	227
8.6.9	Tools of capacity planning	227
8.7	TCO analysis	228
8.8	Post-sizing review	230
	Related publications	233
	IBM Redbooks	233
	Other publications	233
	Online resources	233
	How to get Redbooks	234
	Help from IBM	234
	Index	235

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:

AIX®	InfoSphere™	Resource Measurement Facility™
BladeCenter®	Language Environment®	RMF™
CICS®	Lotus®	S/390®
DB2®	MQSeries®	Service Request Manager®
Domino®	MVS™	System p®
DRDA®	NetView®	System z9®
DS6000™	OS/390®	System z®
DS8000®	Parallel Sysplex®	Tivoli Enterprise Console®
ESCON®	POWER7™	Tivoli®
FICON®	POWER®	TotalStorage®
GDPS®	PR/SM™	WebSphere®
Geographically Dispersed Parallel Sysplex™	QMFTM	xSeries®
GPFS™	RACF®	z/Architecture®
HyperSockets™	Rational®	z/OS®
HyperSwap®	Redbooks®	z/VM®
IBM®	Redpaper™	z/VSE™
IMS™	Redpapers™	z10™
Informix®	Redbooks (logo)  ®	z9®
	Resource Link™	zSeries®

The following terms are trademarks of other companies:

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

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

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

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

You may have several triggers to investigate the feasibility of moving a workload or set of workloads to the IBM® System z® platform. These triggers could be concerns about operational cost, manageability, or delivering the agreed service levels, among others.

Investigating the feasibility of a possible migration or transition to any other platform, including System z, requires a number of basic steps. These steps usually start with an understanding of the current workload and its pain points, and end with a business case to move the workload. It is important to find out how easy a migration is going to be and how much risk will be involved.

In this IBM Redbooks® publication we offer thoughts on how to move through these steps. We also include a chapter with a System z technology summary to help you understand how a migrated workload may fit on the platform.

Our focus in this book is on workloads that are mission-critical and require a high level of availability, including disaster recovery.

The team who wrote this book

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

Alex Louwe Kooijmans is a project leader with ITSO in Poughkeepsie, NY, and specializes in service-oriented architecture (SOA) technology and solutions using System z. He also specializes in application modernization and transformation on z/OS®, both from an architectural and tooling perspective. Previously, he worked as a Client IT Architect in the financial services sector with IBM in the Netherlands, advising financial services companies about IT issues, such as software, hardware, and on-demand strategies. Alex has also worked at the Technical Marketing Competence Center for zSeries® and Linux in Boeblingen, Germany, providing support to clients starting up with Java and WebSphere® on System z. From 1997 to 2000, Alex completed a previous assignment with the ITSO, managing various IBM Redbooks projects and delivering workshops around the world in the area of WebSphere, Java and e-business technology using System z. Prior to 1997, Alex held a variety of positions in application design and development, product support, and project management, mostly in relation to the IBM mainframe. Alex has 23 years of IT experience and has been the lead author of many IBM Redbooks and Redpapers™.

Fernando Gieseler is a System z Technical Sales Specialist in Brazil. He has 12 years of experience in the zSeries field. He joined IBM Brazil just over three years ago. His areas of expertise include every operational system, especially z/VM® and Linux on System z, and design of solutions for database systems. He has written extensively about z/VM and Linux on System z.

Christopher Hastings has more than 29 years of experience as an information architect and writer in the User Experience and Design IBM Systems Group at IBM, Poughkeepsie, NY. He has developed information for the zEnterprise System, z/VM, and WebSphere for z/OS. Chris has an MA in Philosophy from Duquesne University, Pittsburgh, PA.

Stuart Kopel is an IBM Certified IT Specialist working as a Senior Performance Analyst for IBM GTS Services Delivery in Southbury, CT. He joined IBM in 1997 and has over 35 years of experience on MVS™ and z/OS mainframe operating systems, mainly in system performance management. He holds a BA degree in mathematics from the University of Maine and a Masters degree in Business Administration from Clark University. He has also earned the Fellow, Life Management Institute (FLMI) designation from the Life Office Management Association (LOMA).

David Kra is an Enterprise IT Architect, certified by the Open Group at the Distinguished level. His career has been in customer-facing technical sales and consulting roles, advising everyone from CFOs to programmers, operators, and even the cabling team. His expertise spans all layers, tiers, project phases and every communicating programmable platform from IBM over the last 3+ decades, except retail in-store systems. He is a technical negotiator and mediator, innovator, and problem solver. He is known for coming up with solutions to problems that have evaded resolution for a long time. His patents are in e-business and microprocessor design. He is currently employed by Wipro.

James Mitchell is a Staff Software Engineer and IBM Certified System z Specialist with the IBM Worldwide Banking Center of Excellence in Poughkeepsie, NY. He has 25 years of experience in System z operating system development, test, service, and technical sales and marketing support. Over the past five years, he has provided technical consultation and pre-sales support for numerous Financial Services Sector opportunities on System z. He has a BS degree in Computer Science from Jackson State University.

Eduardo Costa de Oliveira, or Eduardo Oliveira for short, is an Executive IT Specialist in the IBM Global Techline Center of Excellence. He serves the role of Techline System z Technology Leader, having recently accepted a position in the SEI/IMF Information Management Foundation growth initiative, as STG System Architect. As an IBMer, he is a 25-year veteran having worked in three different countries, being fluent in Portuguese, Spanish, and English. He sits on the Level 3 and Level 2 Certification Boards, and is a core member of the RACE and FIT for PURPOSE teams. He leads the ZCHAMPIONS TCO WorkGroup team.

Michael Onoufriou is a Client Technical Adviser in Retail Banking supporting two major banks. His 41 years in IBM have been grounded in the data center and has a deep understanding of the qualities of the System z environment and how to deliver the desired retail banking qualities of service on the System z platform.

Dore Teichman is a Sr. IBM Certified IT Architect and Open Group Certified Distinguished IT Architect with IBM. He has more than 25 years of experience in online transaction processing, primarily with ACI Worldwide BASE24 and Tandem, where he served as a regional designated specialist for performance. He holds a degree in Computer Information Systems and Engineering from California Polytechnic University. His areas of expertise are performance engineering, deployment and optimization of high volume online transaction processing systems, and server consolidation. Prior to joining IBM, Dore served as a regionally designated specialist for performance with Tandem Computers, Inc., and is the original author of Guardian Performance Analyzer.

David Ward is a Senior Software Engineer with the IBM Software Group in the United States. He has 30 years of experience in software architecture and design, networking and systems development. His areas of expertise include enterprise messaging and software development methodologies.

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

Ella Buslovich and Alfred Schwab
International Technical Support Organization, Poughkeepsie Center

Martin Jowett
IBM Distinguished Engineer, IBM United Kingdom

Lester Peckover
Enterprise Performance and Capacity Architect, IBM United Kingdom

Kathleen Stalk
Payments Solution Executive

Now you can become a published author, too!

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

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

Comments welcome

Your comments are important to us!

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

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

ibm.com/redbooks

- Send your comments in an email to:

redbooks@us.ibm.com

- Mail your comments to:

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

Stay connected to IBM Redbooks

- Find us on Facebook:
<http://www.facebook.com/IBMRedbooks>
- Follow us on Twitter:
<http://twitter.com/ibmredbooks>
- Look for us on LinkedIn:

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

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

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

- Stay current on recent Redbooks publications with RSS Feeds:

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



Management summary

As software applications are migrated from various platforms, the main focus is on replicating the functionality of the application from the current system to the target system. One area that should not be overlooked is the ability to run the application on a platform that is highly available. In this document, we focus on transitioning the availability requirements of applications to System z from other platforms such as a UNIX distributed environment. The goal is to leverage or improve an existing System z environment to achieve the desired availability—not to replicate the existing architecture. With that thought in mind, you will be able to realize the benefits of having consistent operating procedures such as standardized backup and disaster recovery procedures.

As companies look at moving some of their applications to System z, they are realizing both technical benefits as well as cost benefits for these projects. The movement of applications to System z can be caused by consolidation initiatives due to:

- ▶ The desire to eliminate an island of technology (resources and technology used only for a small number of applications)
- ▶ Standardization of IT operations and development resources
- ▶ Applications running on antiquated hardware
- ▶ Applications running on proprietary software
- ▶ Limitations and high cost of supporting proprietary programming languages
- ▶ Leveraging of existing resources (hardware, software, and labor)
- ▶ Reduction in IT costs (reduction of data center space, power, cooling, and so on)

From a technical perspective, highly available applications have many characteristics that need to be addressed when deciding on the design of the future architecture. From a business perspective the main criteria is the cost and risk factor—that is, you need to determine what the cost is for installing and maintaining an infrastructure to provide the necessary availability for the business. There may be minimal requirements for availability and outage timeframes to reduce any impact on the client. For example, you could build two data centers and have them geographically dispersed to protect against the occurrence of a regional disaster, but the cost to maintain such an infrastructure may increase as you move the data centers further from each other (that is, 100 miles versus 1000 miles of separation).

This is just one example of the decisions that need to be made for all levels of the architecture to provide the proper performance and recovery times while factoring in the proper investments that need to be made.

It should be noted that when transitioning to a System z environment, one should think of leveraging the key characteristics that help define the technical requirements of the highly available infrastructure, which are the Recovery Time Objective (RTO) (time to recover from an outage) and the Recovery Point Objective (RPO) (acceptable amount of data lost); see Figure 1-1. Of course, everyone would desire the RTO and RPO to equal zero (continuous availability) but there are often performance and cost restrictions to accomplish this.

For detailed calculations and a methodology for the cost of availability, refer to *So You Want to Estimate the Value of Availability*, GG22-9318.

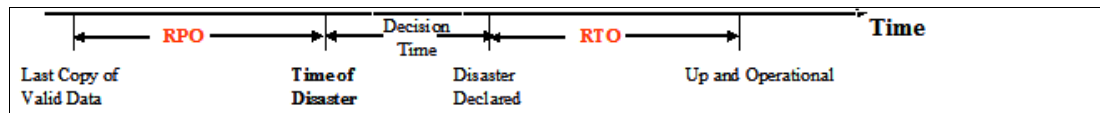


Figure 1-1 RTO/RPO

In this book, we address how to review the current availability requirements, including RTO and RPO, and transform them into the high availability options on System z. Information includes assistance in the following areas:

- ▶ Assessing the current environment
- ▶ Selecting the proper high availability architecture
- ▶ Utilities and tools to achieve high availability
- ▶ Application transition considerations
- ▶ Sizing and capacity planning



Introduction to systems availability

In this chapter we discuss the concept of *system availability*, which we define as the ability of a user to access the system, whether to submit new work, update or alter existing work, or collect the results of previous work. If a user cannot access the system, we say the system is unavailable.

2.1 Introduction

In today's world, people rely on technology for communications, information, location awareness, and other services. Businesses rely heavily on complex computer systems to process, track and manage their critical business operations. Users of all kinds want their systems—wrist watches, cars, airplanes, or computer systems—to be ready to serve them at all times.

Availability is not always improved by adding complexity. Adding more components to an overall system design can improve the system's resistance to failure, but the extra complexity can also reduce it. That is because complex systems inherently have more potential failure points and are more difficult to design and implement correctly. The most highly available systems use a simple design pattern: a single, high quality, multipurpose physical system with comprehensive internal redundancy running all interdependent functions paired with a second, like system at a separate physical location. This classic design pattern is common among many organizations. The same basic design principle applies beyond computing in such diverse fields as nuclear power, aeronautics, and medical care.

2.2 Availability concepts

Availability refers to the system's readiness to accept and process work from its users. It is important to remember that "readiness to accept work" should be considered holistically, that is, a business software system may be active, but it is unavailable if the network that connects it to its users is not functioning.

IBM user groups define *High Availability (HA)* as "the attribute of a system to provide service during defined periods, at acceptable or agreed upon levels, and that masks unplanned outages from users. It employs fault tolerance, automated failure detection, recovery, bypass reconfiguration, testing, problem and change management".

An HA system is designed and implemented to maximize its availability.

Continuous operations is defined by these user groups as "the attribute of a system to continuously operate and mask planned outages from users. It employs nondisruptive hardware and software changes, a nondisruptive configuration, and software coexistence."

Continuous availability is defined as "the attribute of a system to deliver nondisruptive service to the user 7 days a week, 24 hours a day (there are no planned or unplanned outages). It includes the ability to recover from a site disaster by switching computing to a second site".

In this book, we are assuming you are attempting to achieve continuous availability and that you have applications requiring continuous availability, or at least as close to it as possible. You want the service to be available always, with no planned or unplanned service outages. We will consider what it takes to minimize or eliminate the extent and duration of outage even in the case of a disaster taking out a data center. For example, if the network needs to reroute devices from one data center to another, the time to do that is included in the duration of a service outage.

2.2.1 High availability

The term "high availability" usually refers to the system's resistance to *unplanned* failures or outages.

A related concept known as *fault tolerance* describes the system's ability to continue delivering service after one or more of its components have failed. A fault-tolerant system usually has redundancy designed in or added to one or more of its components, such as:

- ▶ Disk subsystems (for example RAID)
- ▶ Processors (for example lock-step processors)
- ▶ Communications (multiple communications ports)
- ▶ Power supply infrastructure

Typically, a system that can achieve 99.99% availability (that is, without an unplanned outage during the defined service interval) is considered highly available.

The key motivation for implementing a highly available system is to minimize (or eliminate) unplanned system outages. This is especially important when the system is supporting a business function that is critical or a government function that has regulatory requirements attached.

2.2.2 Continuous operation

The term “continuous operation” refers to the system's ability to deliver service at all (or some defined) times, even during *planned* outages.

The most common factors that prevent a system from being continuously available are:

- ▶ Software upgrades and reconfigurations
- ▶ Hardware upgrades and reconfigurations

Typically, the need to change or upgrade software or hardware on a single nonredundant system will force periodic planned outages.

For systems that must be in continuous operation, planned outages of this kind are often accommodated by switching workloads to redundant backup components while the primary component is serviced. Achieving the seamless hand-over of the workload to a redundant backup system can be challenging but is a critical success factor for a continuously available system.

Continuous operation places important constraints on the design of application and middleware infrastructure, and these issues are best dealt with during the design phase for these components.

2.2.3 Continuous availability

A *continuously available* system combines the attributes of *high availability* and *continuous operation*; refer to Figure 2-1 on page 6. Such a system is designed so that neither planned nor unplanned outages will interrupt service to users.

Use of component redundancy, good change management and operational procedures are all requirements for successful implementation of a continuously available system.

This goal seems difficult to achieve, because hardware and software components are usually not entirely error-free and maintenance-free, and large computer systems undergo frequent component additions and changes. The solution is to employ hardware components, software, and operational procedures that mask outages from the user. This solution usually requires that recovery from an outage must be performed so quickly that the user does not perceive it as an outage. It also frequently requires the use of redundant components, so that

an alternate component can be used in case of a permanent component failure, or while a component is in maintenance.

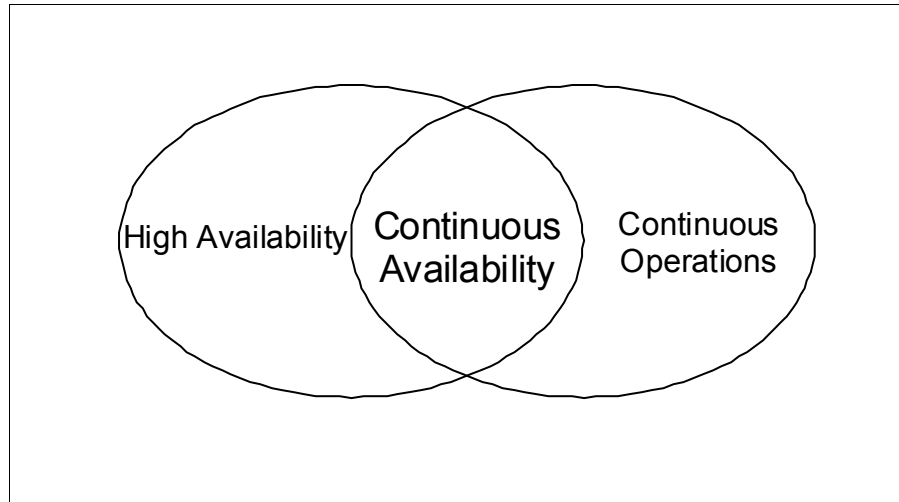


Figure 2-1 Relationship between high availability, continuous availability and continuous operations

2.2.4 Business impact of outages

An unplanned outage of computer systems will result in some business operations not being completed and in some cases, lost. The ultimate financial cost of an unplanned system outage can be realized in a number of ways:

Lost productivity of the system's users

An unplanned system failure might leave people unable to do their work during normal business hours. This can lead to significant delays in the completion of their work and large, unplanned, overtime costs.

Lost productivity of the business's plant or equipment

Factories, production lines and other manufacturing processes can be severely impacted by an untimely failure of a supervisory computer system. Often, this lost productive capacity cannot be recovered.

Unfulfilled customer transactions

Customers of an online service are likely to take business elsewhere rather than wait for service to be restored. These transactions and the business they represent may be permanently lost.

Further, the loss of good will resulting from the failure may lead to future business being lost to competitors.

Lost or incomplete customer transactions

No response might come back for transactions that were already submitted at the time of the outage. In some cases, the client can submit a query immediately or later to see if they were processed. In some cases, if the client resubmits the original transaction, both may get processed, with unintended consequences resulting in client dissatisfaction, calls to customer service, and additional work by humans and technology to compensate for what happened.

Lost computer capacity

The system work that was interrupted may need to be reprocessed when the system is restored. The extra time and computing resources needed to process the work may displace other work.

Additional expense

In some business-to-business flows, when responses do not come back within a defined response time, there is a fee charged to the slow responder. For example, when a bank's response to a card network is delayed, the card network follows prearranged policies as to whether to allow or deny usage of the card. It charges the bank for each request it handles that way.

2.3 Evaluating availability

Four 9's, five 9's, six 9's? What does that really mean?

Availability is measured as the time that the system is available, expressed as a fraction of the total time. More precisely, availability is measured relative to the planned uptime of the system; a system outage during planned downtime does not affect its availability number.

An availability target is often expressed as a percentage, such as 99% or 99.9%, of the planned uptime. Note that when expressed this way, availability is a *probability*, not a determined value. An availability number, such as 99.9999%, does not indicate how much unplanned outage to expect each year. The common statement (see table 5-1 in *A Guide to the ACI Worldwide BASE24-eps on z/OS*, SG24-7684) that "99.9999% availability indicates less than 32 seconds of unplanned outage each year" is simply wrong. The availability percentage is the probability that the service is actually available at any moment in time, during the planned service interval. What 99.9999% availability means is that per 1,000,000,000 planned service hours, one should expect, on average, 1000 hours of unplanned outage.

That is an average number, not always exactly 1000 hours. As long as unplanned outages are random events, availability is a random variable. It has an average and variance around that average.

"99.9999%" availability says nothing about how many or how long outages will be. Having one three-hour outage and 3,600 outages of 1 second each spread randomly over the course of 4,000,000 desired service hours both equate to providing 99.9999% availability. So, more specification is needed. Table 2-1 shows how a yearly availability percentage translates to a monthly and weekly average.

Table 2-1 Availability and average system downtime

Availability	Average downtime per year	Average downtime per month	Average downtime per week
90% (one nine)	36 days	3 days	17 hours
99% (two nines)	4 days	7 hours	2 hours
99.9% (three nines)	9 hours	43 minutes	10 minutes
99.99% (four nines)	1 hour	5 minutes	1 minute
99.999% (five nines)	5 minutes	26 seconds	6 seconds

As an example, 100 outages of 1 second spread randomly over the course of about 28 hours is equivalent to 99.9% availability.

This is an interesting example, because a 1-second outage may not even be perceived by the users as an outage, and still the availability in this example would only be 99.9%! When evaluating actual availability data, an interesting question arises when considering very short gaps in uptime. Should unplanned outages that last only one second be considered when calculating actual uptime for a system? The answer to this question usually lies in the Service Level Agreement (SLA) that binds the system service provider with the user organization.

Brief outages can be regarded as a *delayed* response or *slow* response time glitch if the requests are eventually fulfilled. If the established Service Level Agreement (SLA) states that 99.9% of responses must be within two seconds, and that is being attained, then there was no service disruption and therefore no outage at all.

On the other hand, this might be regarded as an actual outage if the entire second's service requests went unfulfilled. While the SLA may allow, for example, 0.1% of requests to take more than two seconds, it might allow only .0001% to not complete at all. If requests that come during a 1-second period are not processed at all, then that *is* a service outage.

2.3.1 Key availability issues

From a continuous availability perspective, we see concern about:

- ▶ Service outage
- ▶ Non-attainment of Service Level Agreement response time commitments
- ▶ Failure of individual service requests

For each of these, the important questions are:

- ▶ What is the likelihood of having this problem over the life of a system, as designed, including breakage, upgrades, replacement, external events, overload, and moves?
- ▶ How many?

For failure of individual service requests, this is expressed as a fraction or percentage.

- ▶ For how long?

2.3.2 Causes of service outage

Research has shown that hardware failures typically only cause a small portion of unscheduled outages, while applications and IT processes cause by far the most outages. We now briefly mention the most common events causing a service outage.

Application-caused service outage

Application defects that cause every instance of an application to fail are a common source of service outage. Less common are usage patterns that allow some processes of a running application to become deadlocked while contesting a shared resource that they all need.

Processes that cause or allow a service outage

Planned processes such as taking down the only database for nightly backup will obviously not allow for continuous operations. Poor processes allow for service failures. A common example is not preventing running out of disk space for database content or logs.

Breakage that causes a service outage

We think you should really care about something breaking if that breakage will cause a service outage. If the breakage merely causes diminished capacity or performance, that should be acceptable. However, if the system will crash if overloaded, then the breakage will cause an outage unless it is repaired before a peak period arrives that would overload the system.

External events that cause a service outage

This includes power and communications disruptions, fire and water damage, war and civil unrest, severe weather, and specific malicious attacks. Malicious attacks can also come from a source internal to your organization.

Planned changes that require a service outage

For some applications, occasional planned service outages of a few seconds, minutes, or hours are tolerable. Although people often deny it, some “24x365” services are taken down for data center moves, semi-annual software or hardware upgrades, or disaster recovery testing.

Overload that causes a service outage

Some systems are fragile. They have been built such that they will fail when overloaded. Typically, these systems are sized such that they can handle more than the highest anticipated peak moments without reaching saturation. Their normal utilization is often very small.

2.3.3 Service Level Agreement (SLA)

A *Service Level Agreement (SLA)* usually includes expectations of scheduled operations, response time objectives, transaction success rates, disaster recovery *Recovery Time Objective (RTO)* and *Recovery Point Objective (RPO)*, and the change request handling process and responsiveness. These are all IT obligations to the business. Unfortunately, the SLA often does not document the workload to which the SLA applies, in terms of quantity of accounts, transactions per year, peak month, day, hour, or minute. It often does not document what service level is expected when the workload is exceeded.

Breakage that causes SLA non-attainment

The system may have resources that, if missing, do not cause an outage but do diminish capacity. If this happens during peak periods, performance may be unsatisfactory. However, if the system is sized to be able to handle very rare peaks, then most likely the break will happen during a non-peak period and the components can be replaced before the next peak period.

External events that cause SLA non-attainment

Some designs have multiple sites handle the workload concurrently. Capacity and connectivity may be diminished for some period of time when one site is lost due to power and communications disruptions, fire and water damage, war and civil unrest, severe weather, and specific malicious attack.

The SLA may not be satisfied until capacity is increased at the remaining site, some connectivity is rerouted, or the down site brought back into service.

Planned changes that do not cause SLA non-attainment

For some applications, occasional planned service outages of a few seconds, minutes, or hours are tolerable. Although people often deny it, some “24x365” services are taken down for data center moves, semi-annual software or hardware upgrades, or disaster recovery testing.

If the SLA allows for this then the planned change does not cause non-attainment. If the SLA does not, then it would cause non-attainment.

Overload that causes SLA non-attainment

Rather than failing completely, some systems discard service requests when overloaded. Some requests are processed within SLA limits, while others are not processed at all. The SLA criterion for percentage of requests successfully processed will not be met if too many requests are discarded.

Other systems do not discard requests. They maintain maximum throughput, queuing service requests, and handling them as quickly as possible. If the queue gets too long, however, then all requests may end up taking longer than the SLA allows. They might even take so long that the requester is no longer interested in the response.

2.3.4 Service request failure

Breakage that causes a service request failure

In some systems, when a process, processor or other component fails, the particular service request being processed also fails, even though the service as a whole continues to handle other current and future service requests. Depending on the SLA, this may be acceptable.

In a recoverable transactional mode, if a program reads a message and then fails, the message is returned to the queue it came from. Based on requirements, some applications do not need or even must not use this transactional queue model. Queuing systems and applications that do use it must also deal with “poison” messages that always cause service request failures.

External events that cause service request failure

When an external communications failure happens after a service request has been received, the service request may be successful, except that the response cannot be returned to the requester. The application or business process must allow for this possibility. Some transaction processing systems and protocols allow the transaction to be backed out if no acknowledgement of response receipt comes back from the requester.

Also, when a full or partial service outage occurs, service requests being processed at that moment may fail.

Planned changes that cause service request failure

It is unusual for planned changes to cause a service request to fail. If a component is to be taken offline, it can usually be quiesced in a way that stops sending work to it.

Overload that causes service request failure

In an overloaded system that does not completely fail, some requests may be discarded while others complete but the response does not reach the requester in time to be interesting.

Rather than completely fail, some systems discard service requests when overloaded. Some requests are processed within SLA limits, while others are not processed at all. The SLA

criterion for percentage of requests successfully processed will not be met if too many requests are discarded.

Other systems do not discard requests. They maintain maximum throughput, queuing service requests and handling them as quickly as possible. If the queue gets too long, however, then all requests may end up taking longer than the SLA allows. They might even take so long that the requester is no longer interested in the response.

2.4 Other system availability measures

Let us assume that our desired and planned service schedule is 24 hours a day forever. Several commonly used measures are used in this kind of planning and measuring:

Availability The fraction or percentage of time the service is being provided or the component works.

Unavailability The fraction or percentage of time the service is not being provided or the component is not working.

Mean Time to Failure (MTTF)

The average time between service outages or component failures.

Mean Time to Repair (MTTR)

The average time until a service is up again after an outage or until a component is repaired, replaced, or made unnecessary.

Failures in Time (FIT)

The average number of failures expected in 1,000,000,000 hours of desired operation. That could be based on m units operating for n hours such that the total, $m \times n$, is 1,000,000,000 hours.

These are all related by algebra:

- ▶ $\text{Availability} = 1 - \text{unavailability}$
- ▶ $\text{Unavailability} = 1 - \text{availability}$
- ▶ $\text{Availability} = \text{MTTF} / (\text{MTTF} + \text{MTTR})$

Given either availability or unavailability and either MTTF or MTTR, or given MTTF and MTTR, the other two of these four values can be calculated.

$$\text{FIT} = 1,000,000,000 / \text{MTTF}$$

You can analyze the total system design to determine these measures for each category of event: service outage, SLA non-attainment, and service request failure.

2.5 Designing for availability

One instance of infinitely reliable server, storage, operating system, middleware and even applications at one site cannot protect your service from all the hazards described in “Causes of service outage” on page 8.

Even if those components are all failure prone, on the other hand, by using proper architecture, design, implementation, deployment, and processes, it is possible to offer an always available service or one whose service outages, if any, are extremely short.

2.5.1 Elements of a continuously available service

You are using some combination of deployed components and processes to keep your service running. You should include, but not be limited to, the following:

- ▶ Application program and data design that tolerates component breakage, can run multiple instances, handle improper service requests, and scale with additional resources. Application instances can be quiesced, and replaced one at a time rather than all at once.
- ▶ Enhancements to the data model can be made without a total service outage. It is instrumented to facilitate operations and capacity planning.
- ▶ Operation, capacity, and change management processes that keep the service up, upgrade or replace components, monitor for incipient problems, and resolve them without a service outage.
- ▶ Security mechanisms and processes to protect the service from external and internal threats. Component, site, and communications outage testing making sure that failover mechanisms really work.
- ▶ Two or more sites, either sharing the load or with one ready to take over quickly if an active site fails.
- ▶ Multiple server nodes, spread over the sites, with capacity available in case of a server or site outage. They can be serviced, upgraded or replaced without a total service outage.
- ▶ Multiple disk storage subsystems with content replicated between sites by disk, database, or application level replication. They can be serviced, upgraded or replaced without a total service outage.
- ▶ RAID disk level redundancy.
- ▶ WAN capabilities that will reroute traffic in case of a site outage or loss of connectivity. They may also include connectivity from independent providers who share no cable routes above or under ground.
- ▶ LAN capabilities that will reroute traffic in case of a router, switch, cable, network adapter, or server outage. EtherChannel and 802.3ad link aggregation allow for two continuously active paths that normally share the load. It continues with one link in case of an adapter, cable, or switch outage.
- ▶ A Database Management System that allows for backing up, pruning, reindexing, adding space, and so on, without making the content inaccessible. It must also allow itself to be upgraded without a total service outage.
- ▶ Other necessary middleware that scales, clusters and cooperates with or enhances continuous availability.
- ▶ An *Uninterruptable Power Supply (UPS)* for technology and cooling. It may also include power provided from two independent utility substations.
- ▶ Independent cooling control systems, distribution units, and cooling units with sufficient capacity in case one fails.
- ▶ Redundant power supply, including transformers, generators, and oil supply for generators.

2.5.2 Failing over to another site and using moderate quality components, or not

This section presents a strawman minimalist strategy, then explains what is wrong with it in order to clarify what we should do.

A minimalist strategy

A conceptually simple architectural strategy relies heavily on exploiting two sites. Almost any problem at one site can be tolerated by providing all services from the other site until the problem is resolved. Planned outages at one site are handled the same way. You could also use moderate quality components and as few of them as possible at each site.

This approach might be appealing but there may be costs associated with reconfiguring the network to connect all traffic to the remaining active site. Even if both sites normally handle some load, there may be a loss of service accessibility for half the clients while the network is reconfigured or clients reconnect to the other site.

If the second site is passive, not normally handling work, there will be some service outage time while it gets fully up to speed and the network switched over.

Depending on the network and the site's state of readiness, the network switch-over may take longer than any time needed to fully activate the second site. In that case, the network switch-over would mask any negative aspect of having the second site not fully ready all the time. It would also eliminate any added value of having the second site instantly ready.

This minimalist strategy is certainly not suitable if the second site is a passive disaster recovery site provided by a third party that charges large fees for a site activation event on your behalf.

The strategy of using moderate quality components increases how often service requests fail, even if the service remains functional. A deployment built from moderate quality components with as little redundancy as possible would further increase how often failover to the remaining or backup site happens.

For these reasons, most continuous availability solutions try to avoid the need to fail over to another site, especially for planned outages.

2.5.3 Surviving component failure with redundancy

Hardware and software components do break. To avoid a site outage in spite of a component failure, there must be components that can keep the service running in spite of the component that failed. There must be monitoring components that notice the failure. Even better: when the monitoring can notice degraded performance before a failure. Ideally, automated rather than manual operations deactivate the failing component and replace it with a spare. Additionally or alternatively, there may normally active components that can handle the workload.

Depending on the component and how it is used, a service request may fail if it was using the component when it fails. Of course, a failing component must not cause a cascade effect where its failure causes others to fail.

Keeping spares and excess capacity has acquisition, maintenance, power, cooling, space, and often administrative costs. Spare and excess processors also often cause additional software license and maintenance costs.

2.6 Summary

In this chapter we introduced the concept of *systems availability* and the considerations that apply to measuring it, evaluating its effectiveness, and setting appropriate availability goals.

Specifically, the chapter covered:

- ▶ Availability concepts
- ▶ How to evaluate and measure system availability
- ▶ Setting your system availability goals
- ▶ The business impact of system outages

In Chapter 3, “Target architectures for availability” on page 15 we define reference architectures for availability, without going into any technology yet.



Target architectures for availability

In this chapter we introduce the principles of designing an infrastructure environment that delivers on all the requirements of an application being implemented with focus on the qualities of service needed to deliver high availability and effective disaster recovery.

The focus of this chapter is on high availability architecture. Before looking at technology you first need to find out what you require. We discuss applicable technology on System z later in Chapter 4, “System z technology options” on page 49 and present operational models using System z in Chapter 5, “Operational models for high availability” on page 129.

The specific topics discussed in this chapter are:

- ▶ Clustering architectures
- ▶ Activity levels of high availability
- ▶ Factors that determine the approach towards defining the target architecture

3.1 Target reference architectures

This section covers target deployment models from an architectural point of view. It discusses capabilities, their deployment patterns, and the resulting implications for high availability, continuous operations, and disaster recovery. Later chapters map the capabilities to solutions from IBM.

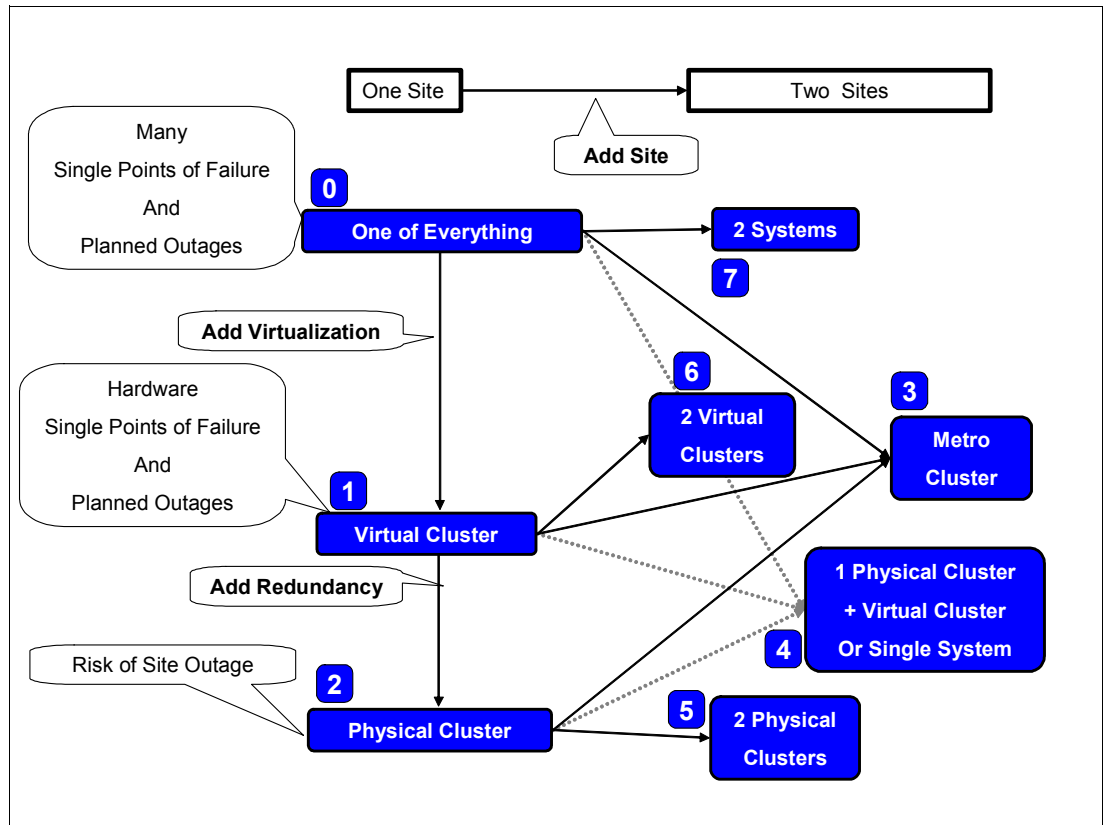


Figure 3-1 System deployment patterns

As depicted in Figure 3-1, we start with a single instance with one of everything. After discussing clustering, we virtualize, and add redundancy.

We then discuss replication between sites and activity levels from dual active to active/cold.

Once we have two sites, we can have, in total, a single metropolitan cluster spread over two locations, one physical cluster with something less at the second site, a physical cluster at each site, a virtual cluster at each site, or one of everything at each site.

In the following sections we discuss how to improve availability with each step.

3.1.1 Each site

The following sections discuss high availability architecture options with one site in mind, or, if you require multiple sites, high availability architecture options in each site.

Single instance with one of everything

Presented only as a starting point, as shown in Figure 3-2, one site with one of everything has limited redundancy, namely what is inside the single server's hardware, disk storage subsystems, and its RAID disk drives.

Even if you add multiple external interfaces such as LAN and SAN adapters, and multiple application instances, it is not a continuous operation configuration.

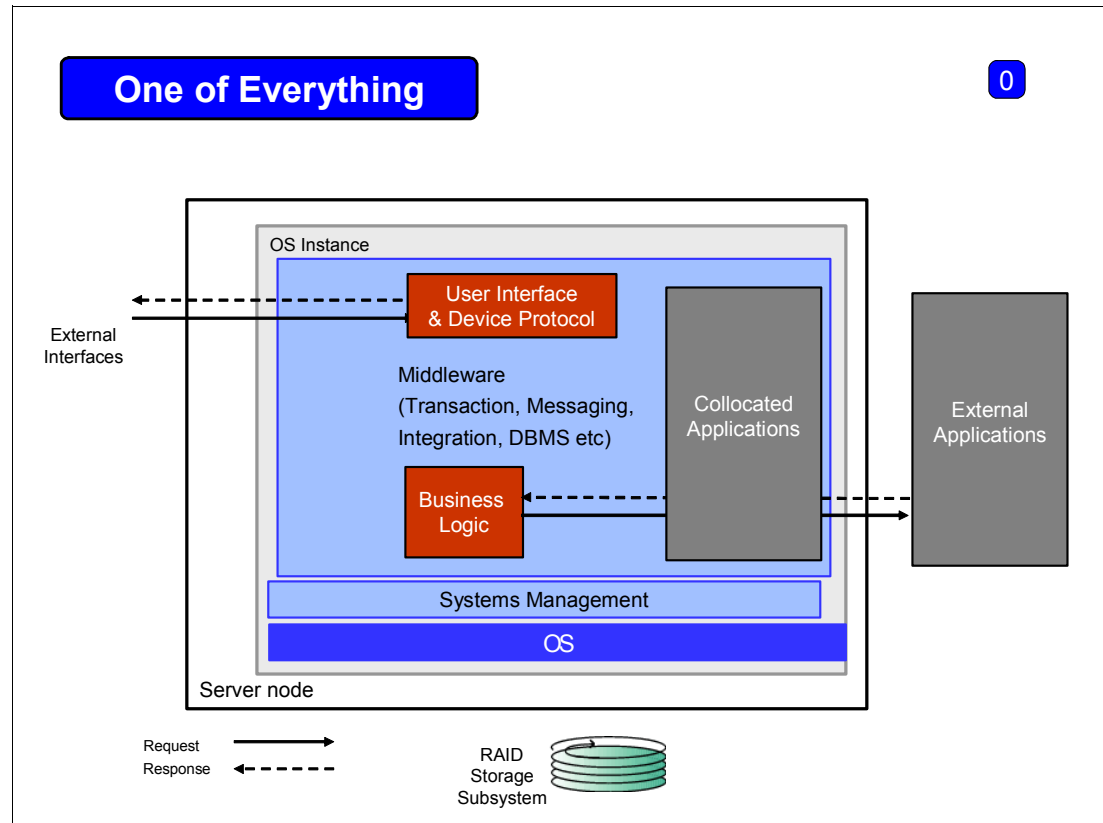


Figure 3-2 Single instance with one of everything

Planned and unplanned software, server, and storage subsystem outages all bring the service down, as depicted in Figure 3-3 on page 18.

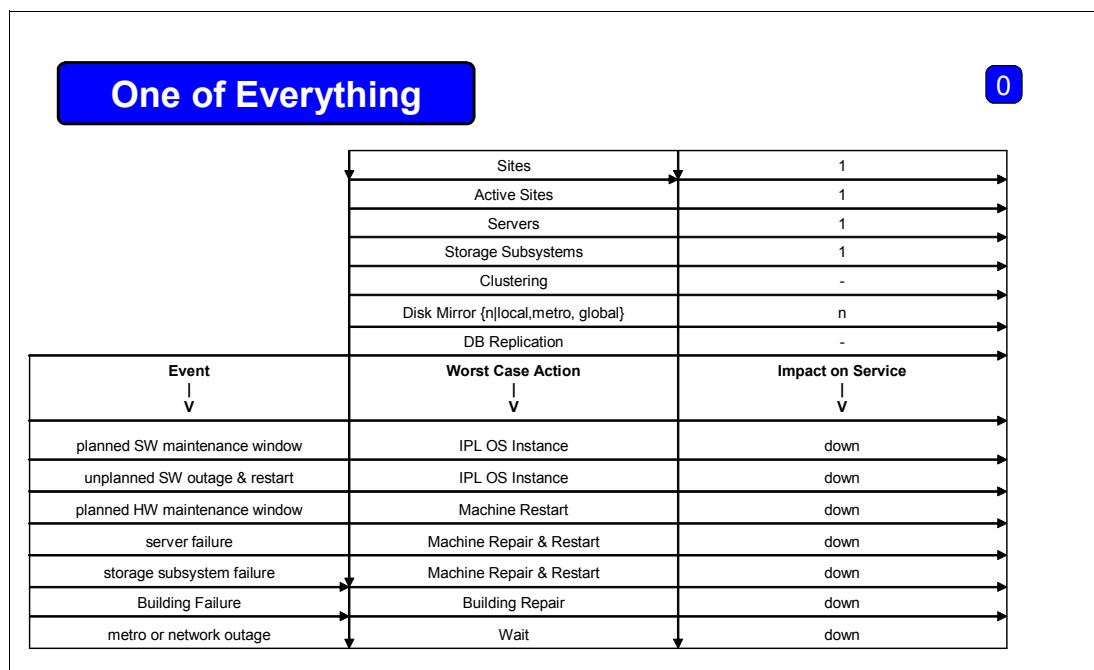


Figure 3-3 Features and availability characteristics of a single instance with one of everything

Clustering

Clustering means adding instances of components for scalability and reliability. This was discussed earlier in 2.5.3, “Surviving component failure with redundancy” on page 13. Figure 3-4 on page 19 depicts two servers that are connected to two clustering devices, two SAN switches, and through them to two storage subsystems, and communications. Connections are duplicated.

The clustering technology can be as simple as direct connect cables between servers, LAN or Infiniband switches, or specialized clustering technology. In some cases the clustering technology holds information shared between the systems. In that case it is important that any information in the clustering device can be recreated in case a clustering device fails. Duplexing of cluster device information can be a hardware function, a software function, or be an application’s responsibility.

The clustering technology can be external or internal to the servers. Figure 3-4 on page 19 shows one example of each. When internal cluster devices are used by server instances in the same server, transfers to and from the clustering technology may be memory-to-memory transfers. In other cases, the internal cluster device is not different from an external device, except that it is physically inside.

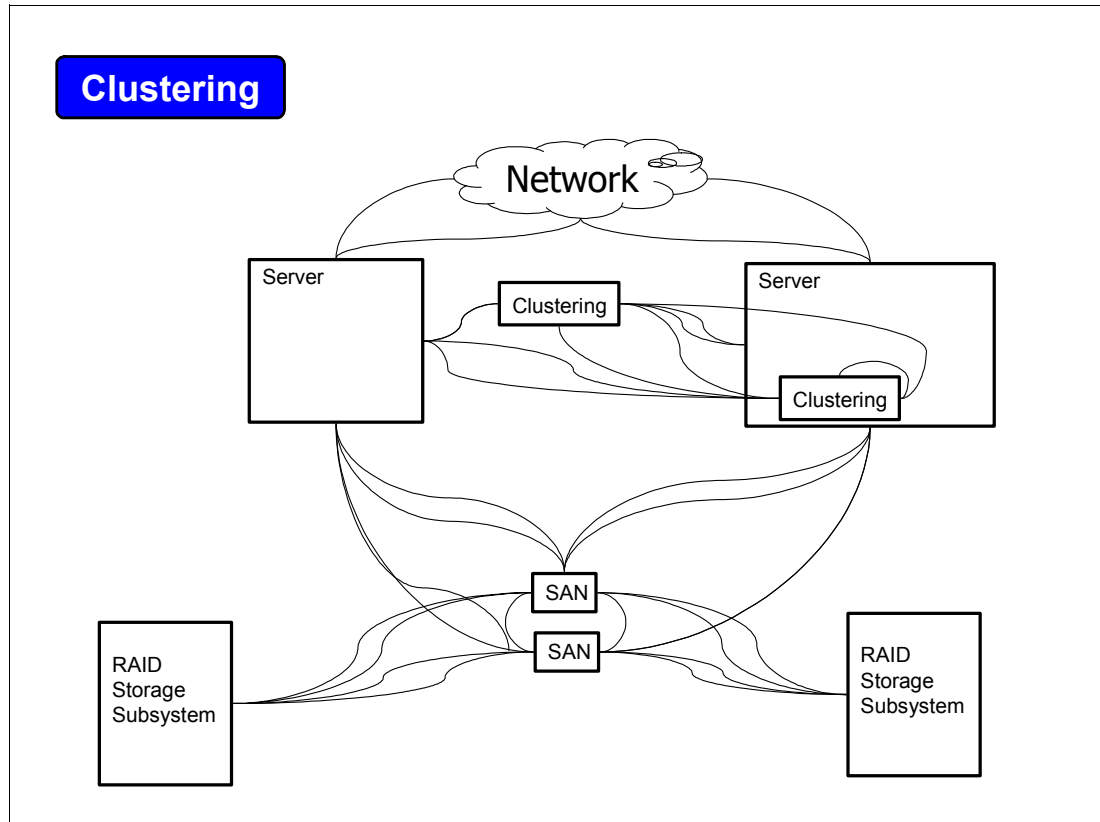


Figure 3-4 High availability cluster components

Three is better than two

The figures in this section mostly show clusters of two physical servers, but there could be many virtual and physical servers in a cluster. For the most critical workloads, the IBM High Availability Center of Competency recommends that there be no single points of failure even when a component is out of service for planned maintenance, upgrade, or replacement. In order to achieve that, a cluster needs to have three servers. If each server's outage is during non-peak periods, then the third server does not need the capacity of the other two. The third server might even be one that is mostly used for other purposes, such as development and test.

Virtualized clustering

Virtualized clustering creates the cluster inside one server. As shown in Figure 3-5 on page 20, there are two virtual servers, also known as logical partitions, and two clustering technology units in the physical server. The clustering technology units themselves could be real or virtual.

Virtual clustering is only feasible under certain conditions:

- ▶ The physical server must be extremely reliable.
- ▶ If there are to be hardware upgrades during the server's years of service, they should not require taking the machine out of service. That may be achieved by a combination of hot pluggability, preinstalled inactive components, and the ability to activate those components without requiring a full server restart.
- ▶ It is feasible and acceptable to switch over to a second site if there is an outage.

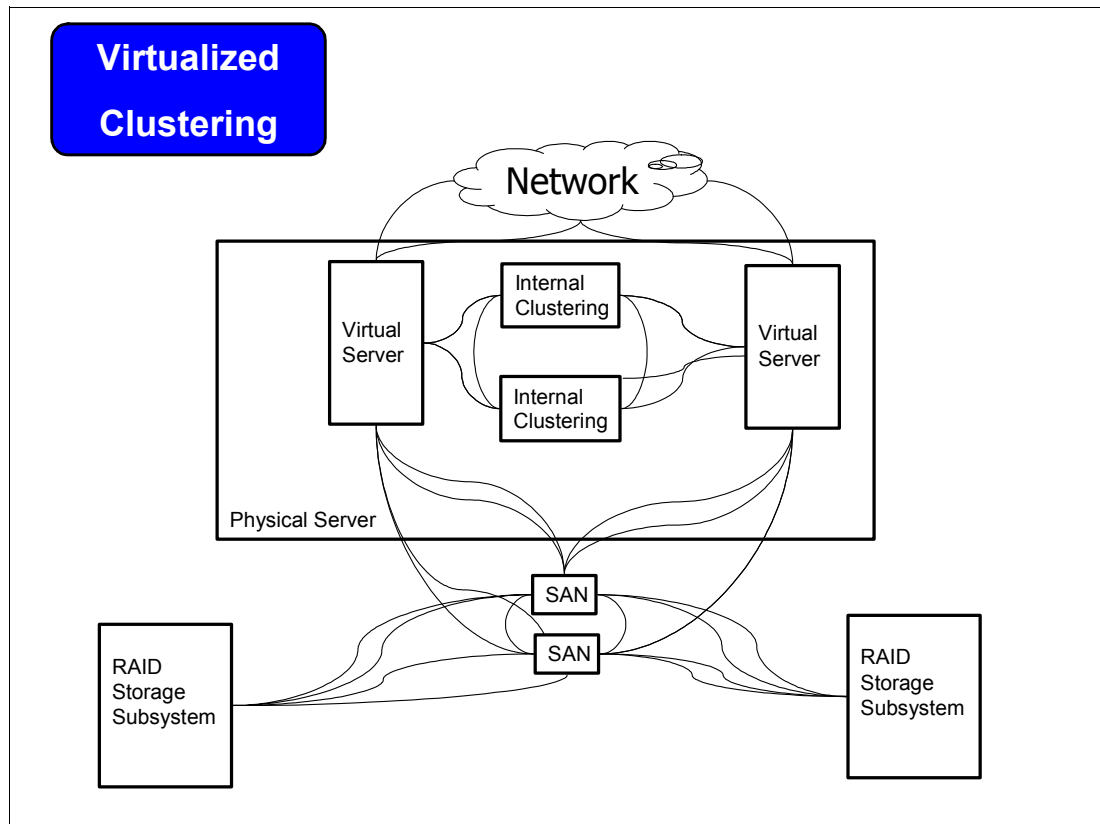


Figure 3-5 High availability virtualized cluster components

Virtual cluster

The *virtual cluster* shown in Figure 3-6 on page 21 is the first step up from a one-of-everything configuration. The service survives the most common causes of outage: planned application, middleware, or operating system outage for maintenance or upgrade. It also survives application, middleware, operating system, or operational process failure within one operating system instance.

If one operating system instance stops providing service, whether due to an application, database, or operating system outage, it stops consuming resources. Where the virtualization technology does logical rather than physical partitioning, those resources can be available to the remaining logical partitions. That gives them the ability to pick up the workload that would have been served by the down instance.

The diagram shows four logical servers in the cluster. Two run the continuous operations applications and others that can be safely colocated with them. Two server instances run applications that prevent continuous operations. These applications, the way they are run, or the way their data is backed up somehow cause planned database, operating system, or middleware outages. If necessary, these servers can be in their own logical cluster.

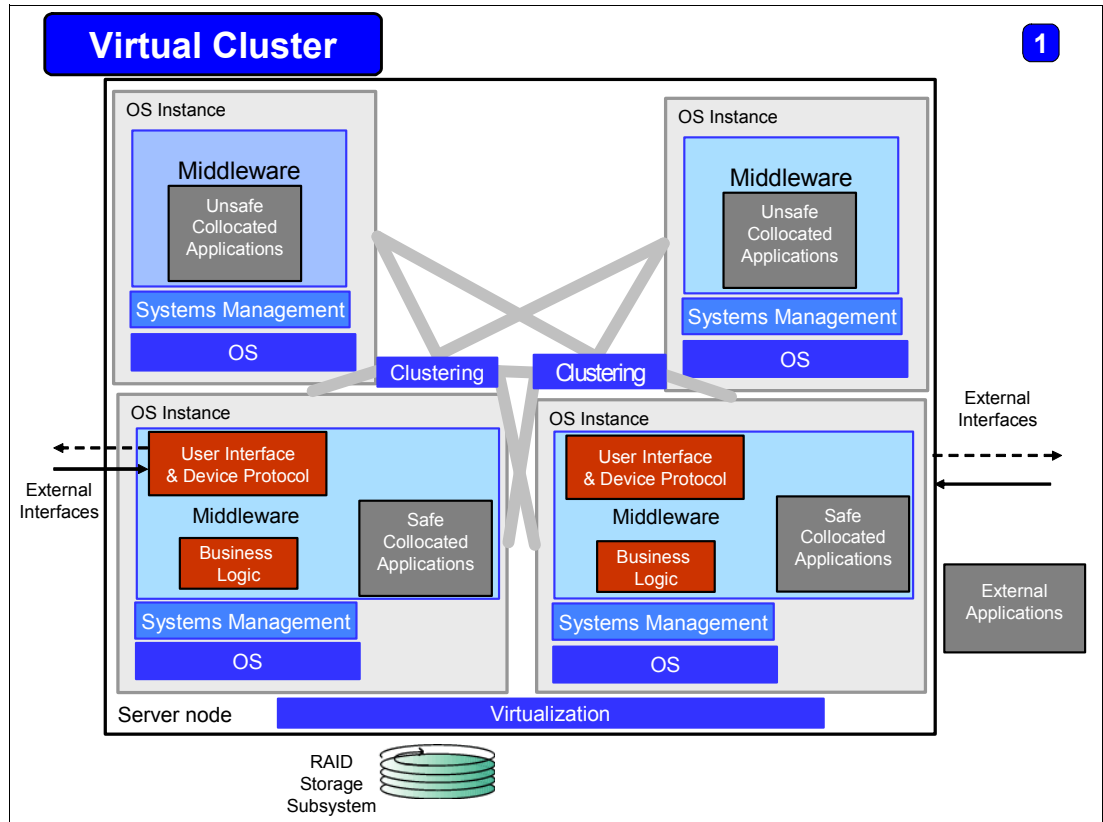


Figure 3-6 Virtual cluster system

As summarized in the table in Figure 3-7 on page 22, the virtual cluster does not provide service during a planned or unplanned physical server or storage subsystem outage. An unplanned software outage, however, may cause in-flight service requests to fail. This is depicted by "No Outage -" in the table.

Virtual Cluster

1

Sites	1
Active Sites	1
Servers	1
Storage Subsystems	1
Clustering	Y
Disk Mirror (n local,metro, global)	n
DB Replication	-

Event ↓ V	Worst Case Action ↓ V	Impact on Service ↓ V
planned SW maintenance window	IPL OS Instance	No Outage
unplanned SW outage & restart	IPL OS Instance	No Outage-
planned HW maintenance window	Machine Restart	down
server failure	Machine Repair & Restart	down
storage subsystem failure	Machine Repair & Restart	down
Building Failure	Building Repair	down
metro or network outage	Wait	down

Figure 3-7 Features and availability characteristics of a high availability virtual cluster

Physical cluster

The next step up is a *physical cluster*. Figure 3-8 on page 23 shows a clustering technology component inside each server. Each logical server is connected to each clustering component. The clustering components are connected to each other.

There are two RAID storage subsystems. Changes to disk drives in one are synchronously replicated to the other. In case of a storage subsystem failure, both servers instantly switch over to the alternate at the hardware level without application or DBMS awareness or operator intervention. Because disk I/O is suspended during the few seconds of switchover, service requests arriving and in process during the switchover may not meet their Service Level Agreement response time objective.

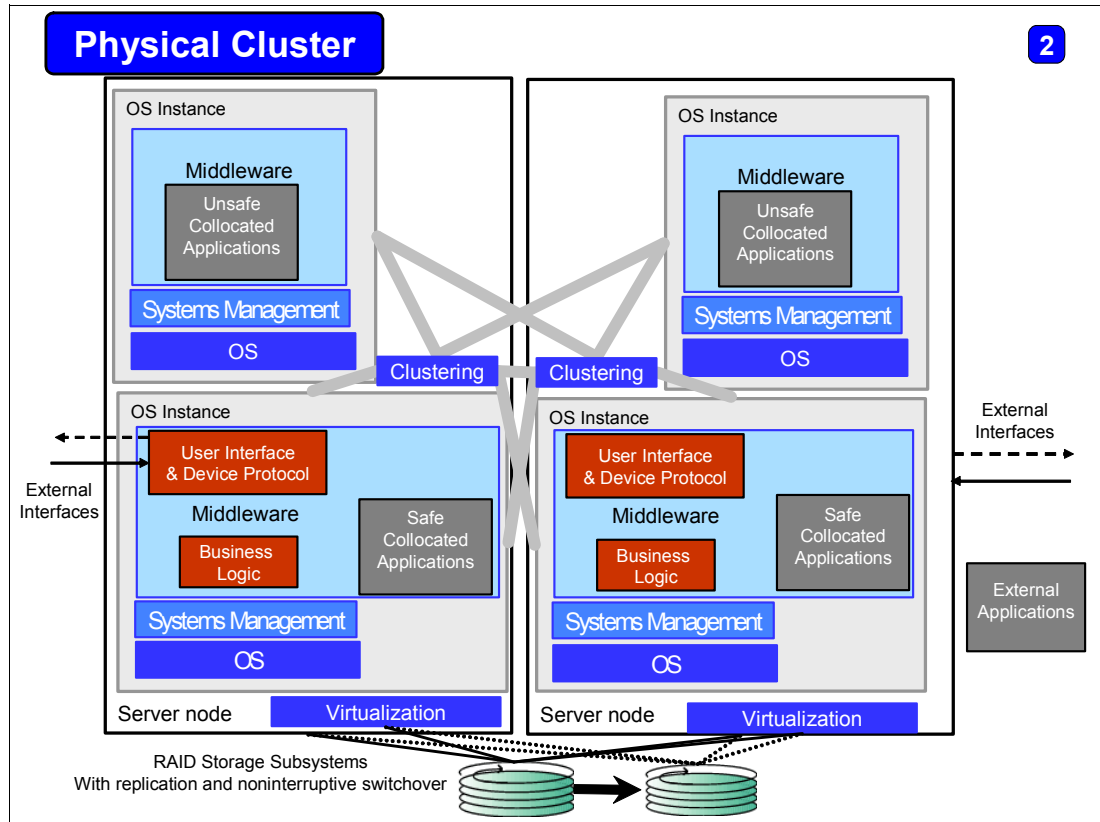


Figure 3-8 Physical cluster system

Except for something that causes the building to be unusable or communications to it to be disrupted, the service is provided in spite of planned or unplanned outage. This is shown in the table in Figure 3-9 on page 24. An unplanned outage, however, may cause in-flight service requests to fail. This is depicted by “No Outage -” in the table.

As mentioned in the section “Clustering” on page 18, if you want to avoid a single point of failure during disruptive maintenance, repair, or replacement, then the cluster needs to have three of each component. With the application still running on merely two of everything during a disruption, an unplanned outage of one of them would not cause a service outage. In reality, this triple redundancy for availability purposes is typically only found in environments such as the space shuttle.

Physical Cluster

2

Sites	1
Active Sites	1
Servers	2
Storage Subsystems	2
Clustering	Y
Disk Mirror {n local,metro, global}	Local
DB Replication	-

Event ↓ V	Worst Case Action ↓ V	Impact on Service ↓ V
planned SW maintenance window	IPL OS Instance	No Outage
unplanned SW outage & restart	IPL OS Instance	No Outage-
planned HW maintenance window	Machine Restart	No Outage
server failure	Machine Repair & Restart	No Outage-
storage subsystem failure	Machine Repair & Restart	No Outage
Building Failure	Building Repair	down
metro or network outage	BC DR Process	down

Figure 3-9 Features and availability characteristics of a physical cluster system

3.1.2 Activity levels for two-site deployments

This activity level discussion is independent of how many systems there are in each site.

When there are two data centers, there are several choices regarding how active and ready each one of them is to handle work and how much work is actually sent to each one. There are many terms that are used, such as “Active/Active” and “Active/Passive”. The terms used in this book are defined here. Definitions for all these are not universally agreed upon in the industry.

Activity level terminology

It may seem that we are splitting hairs by listing so many categories, but each makes a difference versus the others in some of the following: response time, switch-over time, confidence, capacity requirements, conflict resolution, intersite bandwidth, software and hardware involved, acquisition and operational costs.

Dual active

Both sites handle work in this metropolitan deployment. Both share a single set of database content. Both sites can handle work requests arriving at either site. Connections going to one site can be rerouted to the other as quickly as network connections can be rearranged. No conflict resolution is required. Data replication is unidirectional. In case of a primary site storage subsystem failure, both sites switch over to the replica at the second site.

Additional capacity may need to be activated, possibly automatically, to handle peak work loads or one site’s outage. This model, for its level of reliability, is very efficient in terms of total hardware resources required, at the expense of high bandwidth between sites.

Dual active / trickle

A subset of dual active, but one of the two sites normally only handles a tiny fraction of transactions. This demonstrates its readiness to take

over as needed. Hardware and software costs at the second site are lower than in regular dual active.

- Dual active / ready** A subset of dual active / trickle, but all transactions are normally handled by one site. The application at the ready site is up and running and waiting for all the network connections to be switched over to it.
- Dual active / warm** Similar to dual active/ready, but the database and application are not up and running at the warm site. They are ready to be started. The warm site can be further from the active site as compared to the other dual active deployments. At the second site, the application and middleware do not hold any database or disk level locks.
- Independently active** Application instances and databases at both sites handle work completely independently. Data from each site is replicated asynchronously to the other for business continuity and disaster recovery purposes only. There is no distance limitation.

There are special implications if the application has statefulness, whether during a usage session or over the long term, such as account data. In that case, requests during a session or for an account must always go to the same location.

If and when a site comes down, an additional instance of the application and its DBMS are brought up at the remaining site. That additional instance does the work that would have been done at the down site. It uses the data that had previously been replicated to there.

- Active / active** Both sites handle work. Each site has its own database content, which *somehow* includes information about transactions at both sites. The *somehow* is discussed in 3.1.3, "Information replication between sites" on page 26. Connections going to one site can be rerouted to the other as quickly as network connections can be rearranged. Active/active may require conflict resolution at the application and/or database level.
- Active / trickle** A subset of active / active, but one site normally only handles a tiny fraction of transactions. This demonstrates its readiness to take over as needed. Additional capacity may need to be activated to handle peak work loads, possibly automatically.
- Active / ready** A subset of active / active, but all transactions are normally handled by one site. Data replication is unidirectional. The database and possibly the application at the ready site are up and running and waiting for all the network connections to be switched over to it. Conflict resolution is not required.
- Active / warm** Similar to active / ready, but the database and application are not up and running at the warm site. They are ready to be started.
- Active / cold** All work is done at one site, while replication copies data at the disk level to the cold site. Depending on the replication technology, the cold site's computers might even be powered off.

In all cases, as a site's role goes from active to trickle to ready to warm to cold, the amount of active hardware and software diminishes. That saves ongoing power. Depending on technology and vendor terms and conditions, each step may enable dramatic savings in acquisition and support costs for both hardware and software.

3.1.3 Information replication between sites

This information replication discussion is independent of how many systems are in each site.

Information needs to be replicated between sites for two reasons:

- ▶ Disaster recovery in case of a site outage
- ▶ Intersite consistency when both sites normally handle service requests

Information can be replicated by technologies that operate at disk, database, and application levels, each with its own characteristics and implications.

Disk

With disk level replication, the disk subsystem at one site receives changes made at the disk subsystem at the other site. This can be applied to whichever disks need to be replicated, independent of any particular application, access method, or DBMS.

Replication may be synchronous with its distance limitations, increased application response time and bandwidth requirements. This is used in all the variations of dual active as shown labeled with a (1) in Figure 3-11 on page 28. The label (0) does not represent replication. It represents normal usage. Both sites normally use the disk subsystem at site A. In case of a Site A storage subsystem outage, both servers switch over instantly to Site B's replica. That usage is labeled (0').

Replication may be asynchronous without distance limitations or impact on response time and with more flexible bandwidth requirements. However, with asynchronous replication, the information at the target receiving site will lag somewhat behind what is at the source sending site.

Synchronous or asynchronous disk replication may be used in Independent Content replication as shown labeled with a (1) or a (4) in Figure 3-13 on page 30.

The replica is at the target site for disaster recovery purposes only. While it is receiving changes from its source site, computers at the target site cannot write to the replica disks, and should not read directly from them. Target site computers can read from a moment-in-time copy of the replicas.

Some techniques use a program on a server at the receiving site. Typically, it asynchronously pulls changed data from the source site disk subsystem and writes it to the target disk subsystem. This is labeled with a (4) in Figure 3-13 on page 30 where the program is called a "data mover."

The disk subsystem sends the changes to the target site. When a server has many SAN adapters, disk writes that have been queued up may complete in an order different from the sequence of writes in an application. This might happen for other reasons also. This could lead to the state of the storage subsystem being different from the server's.

If you value having the target disk copy representing a consistent state, use replication technology that can make sure that the state of the replicated disk subsystem is consistent with a moment in time as perceived by the servers at the source site. That often requires storage consistency technology in both the server and the storage subsystem.

Database

Database level replication moves committed database changes for the tables you specify. As a result, it happens asynchronously after a transaction has completed or at least committed its changes.

SQL-based replication works based on changed rows without necessarily considering the semantic interrelationships among tables. Log-based replication sends the changes for a transaction as a unit, maintaining semantic consistency. Both techniques are widely used.

When there are normally no local changes at the target site, we call the replication unidirectional. Two unidirectional streams are labeled (2) in Figure 3-13 on page 30. Even though these streams are flowing in opposite directions between two sites, each is unidirectional. There is no normally active application at each site updating the recovery target copy.

In active/active and active/trickle activity modes the application at each site is updating its local database. Furthermore, the database at each site must include information about state changes at the other site. This provides both sites an active unified view of the service's persistent data. Bidirectional database replication can accomplish that.

However, replication into an active database exposes the risk of conflicts or collisions. This can happen when services executing at both sites update or increment the same fields in equivalent rows. If your `account_balance` is \$500, and a transaction at one site credits your account with \$100 while almost simultaneously a transaction at the other site credits your account with \$200, you want the final `account_balance` to be \$800, not \$700 and not \$600.

The replication technology must notice the conflict. Based on your specification, the technology must handle the conflict, invoke your logic to handle it, or ignore it.

Bidirectional database replication with conflict resolution is labeled (2) in Figure 3-12 on page 29.

Database level transaction replication avoids the need to re-execute the transaction's logic and integration flows at the target site.

Database level replication does not deal with replicating system objects or flat files between sites.

Application

Application level replication requires that the application at each site send transaction data to its peer. That could be a duplicate of the incoming service request, or a notification containing necessary results from a completed transaction.

Two unidirectional application level replication streams are labeled (3) in Figure 3-13 on page 30. The active source systems are at the top left and bottom right, colored in green. The target inactive systems are at the top right and bottom left, colored red, labeled with *italic text*.

Either way, conflict resolution logic is required if the target application is also handling service requests that change service or account state. This bidirectional application level replication is labeled (3) in Figure 3-12 on page 29. The target must re-execute the service request to some extent or apply state changes resulting from the source site's execution.

Application level replication does not deal with replicating system objects, database tables, or flat files between sites. It does not replicate any changes to the application's database made by utilities or directly by authorized administrators.

Single DB metropolitan clustering data replication

Figure 3-11 on page 28 depicts disk level synchronous replication, labeled (1), from one active site to another in the same metropolitan area. The DBMSs at both sites are actively sharing the disk subsystem at site A, as depicted by lines labeled (0). They do not know of the

existence of the replica at site B. Active technology is shown in green. The target replica is inactive, colored red, and labeled in *italics*. See Figure 3-10 for an explanation of the colors.

While site B's server, including its clustering technology, is active and shown in green, the DBMS and application might not be. Depending on whether the deployment is dual active, dual active/trickle, dual active/ready, or dual/active/warm, the appropriate color might be green, yellow, orange, or red. The diagram uses a color in the middle, gold.

If Site A completely fails, site B's server hardware switches over to use the replica in site B. That usage is shown with a red broken line and is labeled (0'). If Site A's disk subsystem fails, the servers in *both* sites switch over to the replica disk subsystem in site B. Those usages are shown with red broken lines labeled (0') and (0''). If both sites are functional, but communications is broken between them, then site A is left running and site B is taken down.

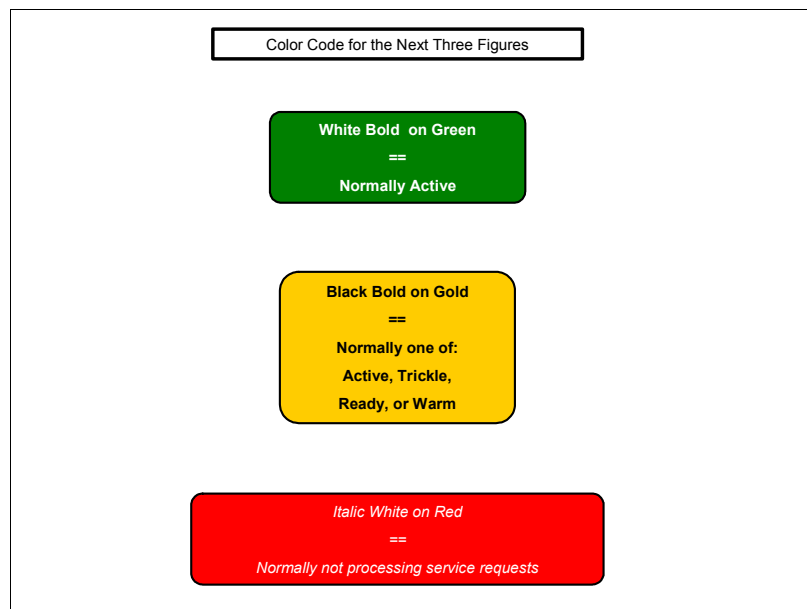


Figure 3-10 Color code for the next three figures

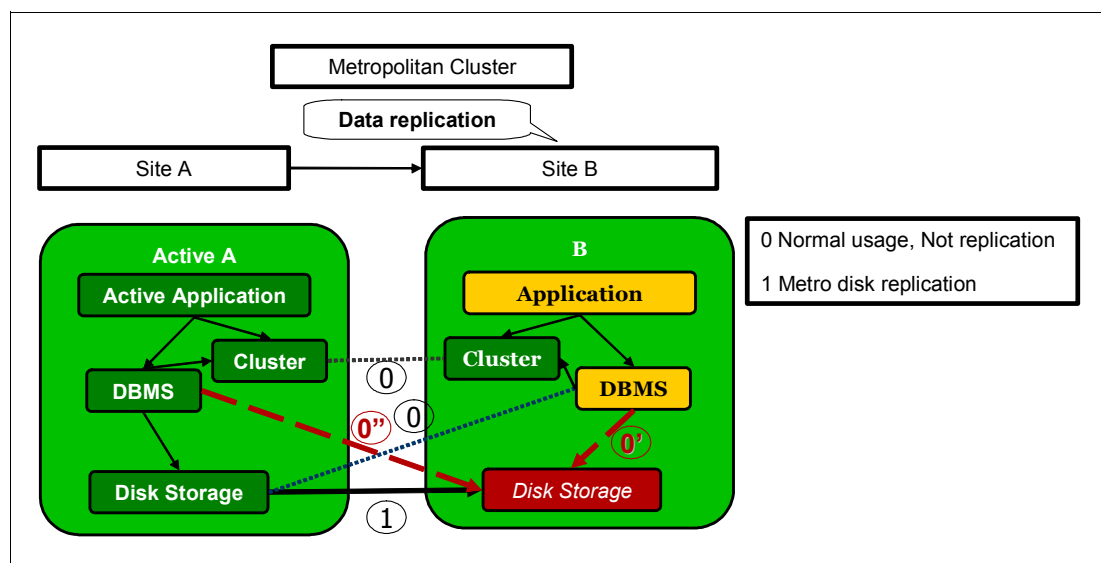


Figure 3-11 Single DB metropolitan clustering data replication

Coordinated-content DBMS or application-level replication

When two sites are active to some extent, and have separate DBMSs, but the DBMSs need to reflect the aggregate set of activities, then the database content must be replicated in a coordinated way, including dealing with conflicts. This can be done at the application level or the database level.

DBMS level replication

Figure 3-12 on page 29 shows two sites, with systems and the DBMS running at both sites. The application at site A is active. The line marked (2) shows database-level replication sending changes over to site B. Ignore the presence of the line marked (3).

If the application at site B is handling any requests, then the overall deployment is running active/active or active/trickle. In that case, database replication would be bidirectional and therefore conflict resolution must be active at both sites. According to the color code, the application at site B and conflict resolution at both sites all change from orange to red.

Application-level replication

In the same Figure 3-12 on page 29, with the application active at site A, the line marked (3) shows application-level replication sending changes over to site B. Ignore the presence of the line marked (2). If the application at site B is handling any requests, then the overall deployment is running active/active or active/trickle. In that case, application-level replication would be bidirectional and therefore conflict resolution must be active in both sites. According to the color code, the application at site B and conflict resolution at both sites all change from orange to red.

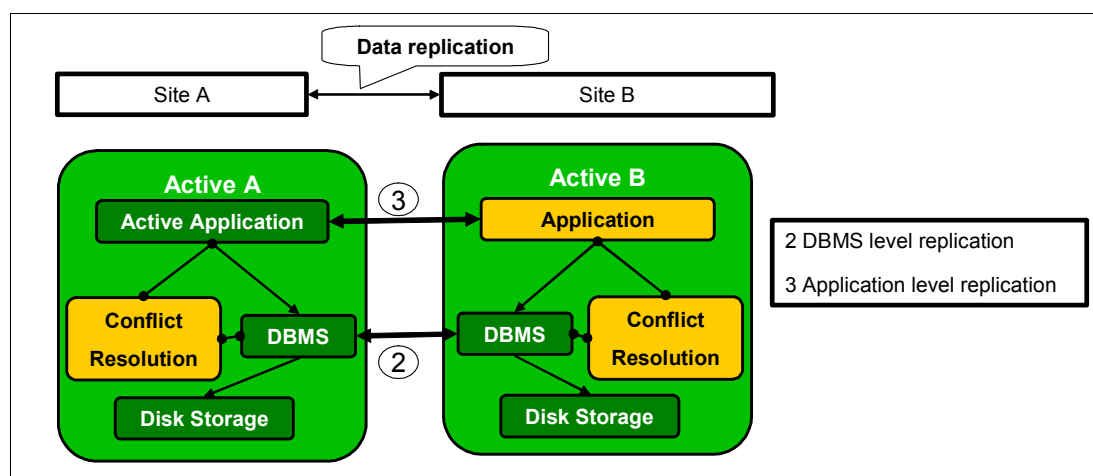


Figure 3-12 Coordinated content replication

Independent-content replication

The Independently Active deployment model has active systems at each site normally handling requests on behalf of its own geography, account group or some other segmenting factor. If one site fails, the other site must take over, but will do so with another instance of the application and its database representing the failed site. Information needs to be transmitted between sites, but not merged.

Figure 3-13 on page 30 shows that any form of replication can be used, whether disk, disk with data mover, DBMS, or application-based. The green active set at each site does productive work. The red passive set at each site catches data coming from the other site to make a replica.

When using any disk level replication, labeled (1) and (4), the passive DBMS and application would not be up at all. When using DBMS-level replication, labeled (3), the passive application need not be up at all. There is no need for any conflict resolution function.

For clarity, Figure 3-13 on page 30 shows separate active and passive servers, application instances, storage subsystems, and DBMSs at each site. In reality, the active and passive elements can run in the same physical server(s), cluster, virtual server, DBMS, and storage subsystem.

If using database or application level replication, one running DBMS instance can independently support both sites' databases. In case of a site outage, that running DBMS can support new active application instances representing the failed site.

If using disk-level replication, in case of a site outage, the passive disks would be made available to the local system, and a DBMS instance started, possibly in a newly started logical server.

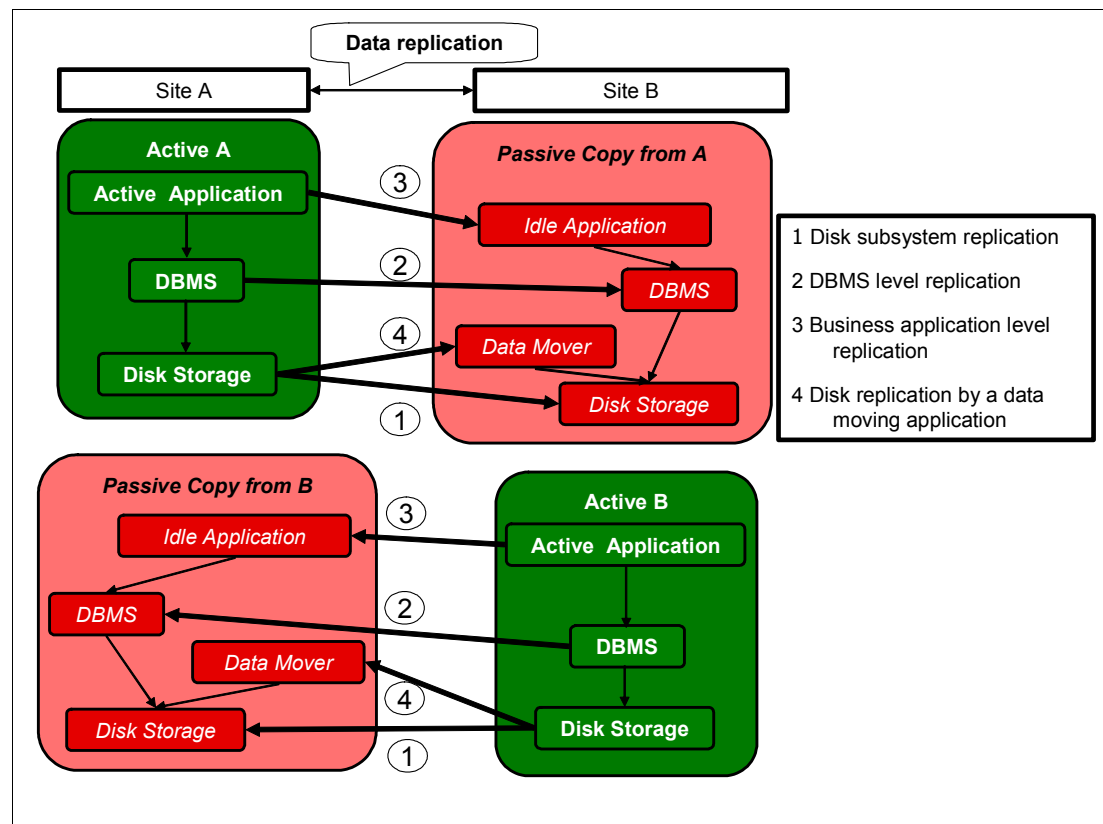


Figure 3-13 Independent content replication

Summary of activity levels and replication techniques

The table in Figure 3-14 on page 31 shows the activity levels, site outage event handling, their *recovery time objective (RTO)*, *recovery point objective (RPO)*, and which replication technologies can be used with that activity model.

Two Site Activity Levels: Event Handling, RTO, RPO, and Replication Techniques			Replication Technique -->	Disk	Data Mover	DBMS	Application
Activity Level	Event Handling	RTO	RPO				
Dual Active	Switch NW/2	Half NW	0 seconds	Y			Y
Dual Active / Trickle	Switch NW	NW	0 seconds	Y			Y
Dual Active / Ready	Switch NW	NW	0 seconds	Y			Y
Dual Active / Warm	Switch NW; Activate DBMS & Application	minutes	0 seconds	Y			
Independently Active	Switch NW/2; Activate anything idle	Half NW to minutes	Synch: 0 seconds Asynch: low seconds	Y	Y	Y	Y
Active / Active	Switch NW/2	Half NW	low seconds			Y	Y
Active / Trickle	Switch NW	NW	low seconds			Y	Y
Active / Ready	Switch NW	NW	low seconds			Y	Y
Active / Warm	Switch NW; Activate DBMS & Application	minutes	Synch: 0 Asynch: low seconds	Y	Y		
Active / Cold	Switch NW; Power Up; IPL; Activate DBMS & Application	minutes	Synch: 0 Asynch: low seconds	Y			

Figure 3-14 Two site activity levels: Event handling, RTO, RPO, and replication techniques

3.1.4 Multisite deployment

The activity levels and replication techniques described in Figure 3-14 are independent of any clustering or storage mirroring within each site. Now we will examine aggregate two-site deployments.

Metropolitan cluster

The *metropolitan cluster* shown in Figure 3-15 on page 32 looks very similar to the single-site physical cluster shown in Figure 3-8 on page 23. It should. They are the same, except that the server and storage subsystem on the left are in one building and the server and storage subsystem on the right are in another, up to about 25 km apart. This allows both sites to access one active disk storage subsystem, and for that subsystem to replicate synchronously to its peer at the second site.

The metropolitan cluster runs the dual active family of activity levels. Disk level replication is used for both disaster recovery and a disk storage subsystem outage. Application instances that share context information in shared memory, shared queues or shared files do that at the application-to-application level through the clustering technology.

The dual active/ready and dual active/warm activity levels support greater distances between sites. At the dual active/ready level there are no active usage requests going between sites. At the dual active/warm level there is even less cross-system traffic.

In case of a disk storage subsystem failure, all the servers switch over to using the alternate.

In case of a site outage at the site with the active disk storage subsystem, the other site alone switches over to using the alternate disk subsystem.

Admittedly, this deployment pattern requires a lot of duplexed bandwidth between sites. Very high-speed connections are required between the disk storage subsystems and from them to the servers. Similarly, the clustering technology instances are connected to each other and to the servers at both locations.

Duplexed wave division multiplexers can reduce the number of fibers between the sites. Multifiber cables can reduce the number of intersite cables, but preferably not down to just one. Ideally, the capacity for each connection is shared over cables that go over completely different physical routes between the sites. That way, there is no communication failure when a trench is dug, a pole is knocked down, or there is an accident at a fiber junction point.

Adding another server to the cluster at one or both sites can make a big difference. As depicted, the multisite cluster relies on one site to take over completely for planned and unplanned physical server outages at the other site. If the cluster is expanded to include two servers at either or both sites, that takeover would no longer be necessary for the site(s) with two servers. The second server at each site could be an older model, possibly with inactive capacity that can be activated if and when needed.

In summary, with two servers and two disk storage subsystems, the metropolitan cluster(Figure 3-16 on page 33) provides continuous availability for multiple cluster-enabled applications.

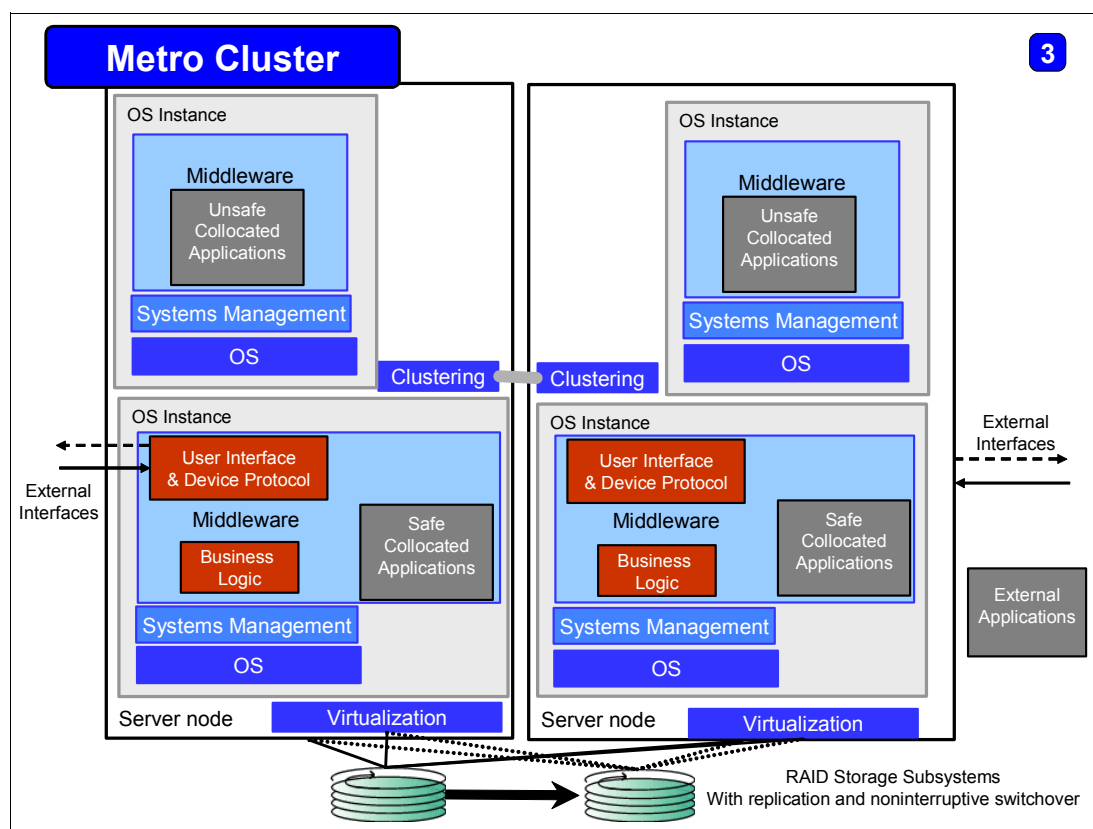


Figure 3-15 Metropolitan cluster

Metro Cluster

3

Sites	2
Active Sites	1..2
Servers	2..4
Storage Subsystems	1/site
Clustering	Y
Disk Mirror {n local,metro, global}	metro
DB Replication	-

Event ↓ V	Worst Case Action ↓ V	Impact on Service ↓ V
planned SW maintenance window	IPL OS Instance	No Outage ; partial network switchover
unplanned SW outage & restart	IPL OS Instance	No Outage- ; partial network switchover
planned HW maintenance window	Machine Restart	No Outage ; partial network switchover
server failure	Machine Repair & Restart	No Outage- ; partial network switchover
storage subsystem failure	Machine Repair & Restart	No Outage- -
Building Failure	Building Repair	No Outage- ; partial network switchover
metro or network outage	BC DR Process	No Outage- ; partial network switchover

BC DR Recovery Time typ ; max	Partial network switchover
BC DR Recovery Point typ ; max	0

Figure 3-16 Features and availability characteristics of a metropolitan cluster

One physical cluster plus a virtual cluster or single system

This deployment model is extremely popular. As shown in Figure 3-17 on page 34, site A has a physical cluster with two or more servers. Site B has one server, set up as a single logical system or as a virtual cluster. Typically, data replication between sites is at the disk subsystem level. If necessary, database-level or application-level replication is added for specific applications.

Typically, the server at site B is not doing production business work. If it is powered up, with an OS instance or virtual cluster running, it most commonly does development, test, and data archiving.

The features and availability characteristics of this model are shown in Figure 3-18 on page 34.

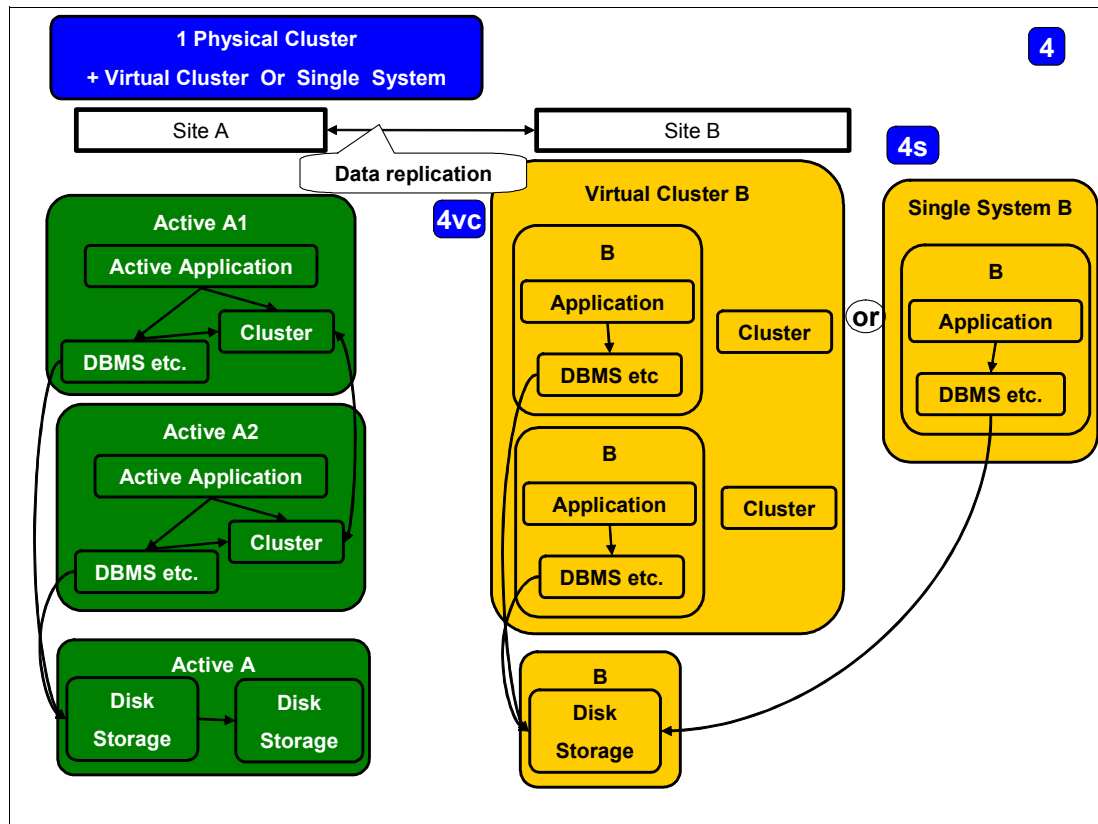


Figure 3-17 One physical cluster plus a virtual cluster or single system

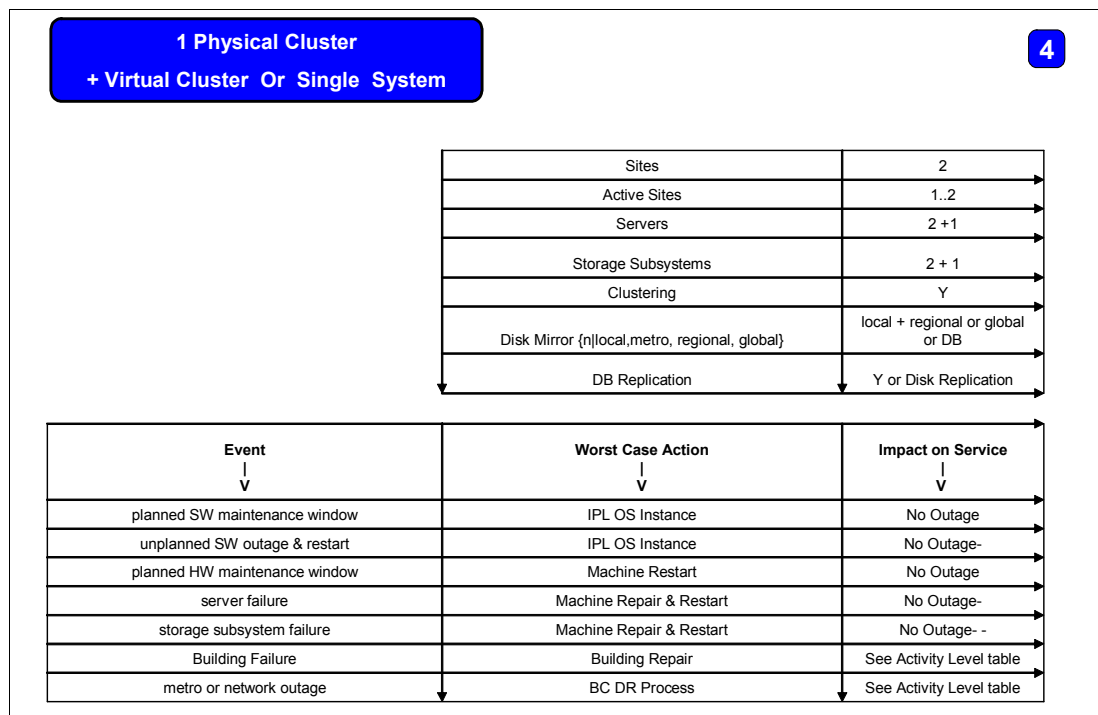


Figure 3-18 Features and availability characteristics of one physical cluster plus a virtual cluster or single system

For the actions, RTO, and RPO for cases in the table in Figure 3-18 on page 34 where it says “See Activity Level table,” refer to Figure 3-14 on page 31.

Two physical clusters

There are several situations that give rise to having a physical cluster in each of two sites, as shown in Figure 3-19 on page 35. Each data center is active and located based on regional business and communications needs, contractual obligations, privacy and transborder data flow laws, collocated staff, or the result of mergers and acquisitions.

Furthermore, the applications running in each site need the capacity or reliability of a physical cluster. That being the case, the scenario also includes local mirroring of the disk subsystem. That way neither a disk subsystem nor server outage can take the site down.

The features and availability characteristics of two physical clusters are shown in Figure 3-20 on page 36.

With two physical clusters, the only reason to shift one site’s work to the other is a site outage or the loss of communications to the site. Assuming an application is running at site A, the level of activity for that application at site B could be anything from cold to active. That is why site B is shown in gold, which is between green and red in the spectrum.

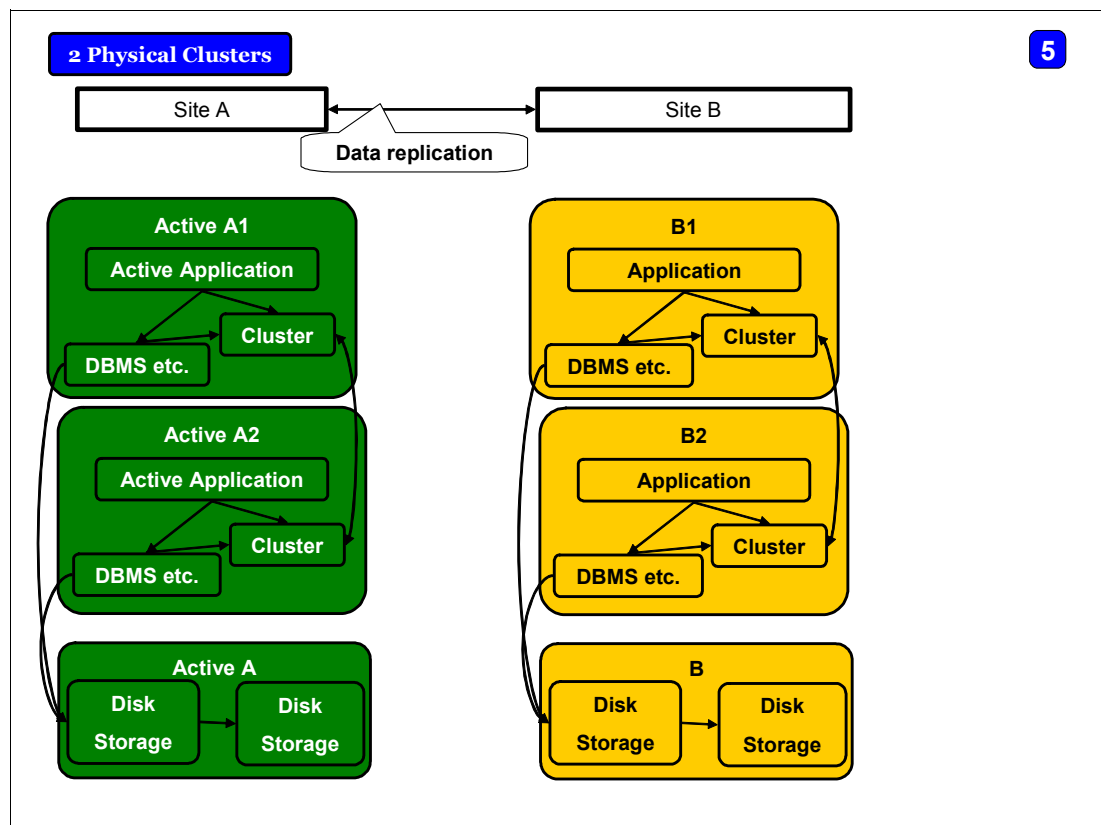


Figure 3-19 Two physical clusters

Sites	2
Active Sites	1..2
Servers	4
Storage Subsystems	2 + 2
Clustering	Y
Disk Mirror {n local,metro, regional, global}	local + regional or global or DB
DB Replication	Y or Disk Replication

Event ↓ V	Worst Case Action ↓ V	Impact on Service ↓ V
planned SW maintenance window	IPL OS Instance	No Outage
unplanned SW outage & restart	IPL OS Instance	No Outage-
planned HW maintenance window	Machine Restart	No Outage
server failure	Machine Repair & Restart	No Outage-
storage subsystem failure	Machine Repair & Restart	No Outage- -
Building Failure	Building Repair	See Activity Level table
metro or network outage	BC DR Process	See Activity Level table

Figure 3-20 Features and availability characteristics of two physical clusters

For the actions, RTO, and RPO for cases in the table above where it says “See Activity Table,” refer to Figure 3-14 on page 31.

Two virtual clusters

If it is acceptable to switch work to the other site for some planned or unplanned events, typically for a few hours twice a year, then having a virtual cluster at each site may be feasible. This is shown in Figure 3-21 on page 37.

Assuming an application is running at site A, the level of activity for that application at site B could be anything from cold to active. That is why site B is shown in gold, which is between green and red in the spectrum.

This model is safe from planned and unplanned software outages. It may be hard to justify two physical servers when server unplanned outage intervals are on the order of a human lifetime and when fewer and fewer hardware maintenance and upgrade events require taking the entire server down.

The features and availability characteristics of two virtual clusters are listed in Figure 3-22 on page 37.

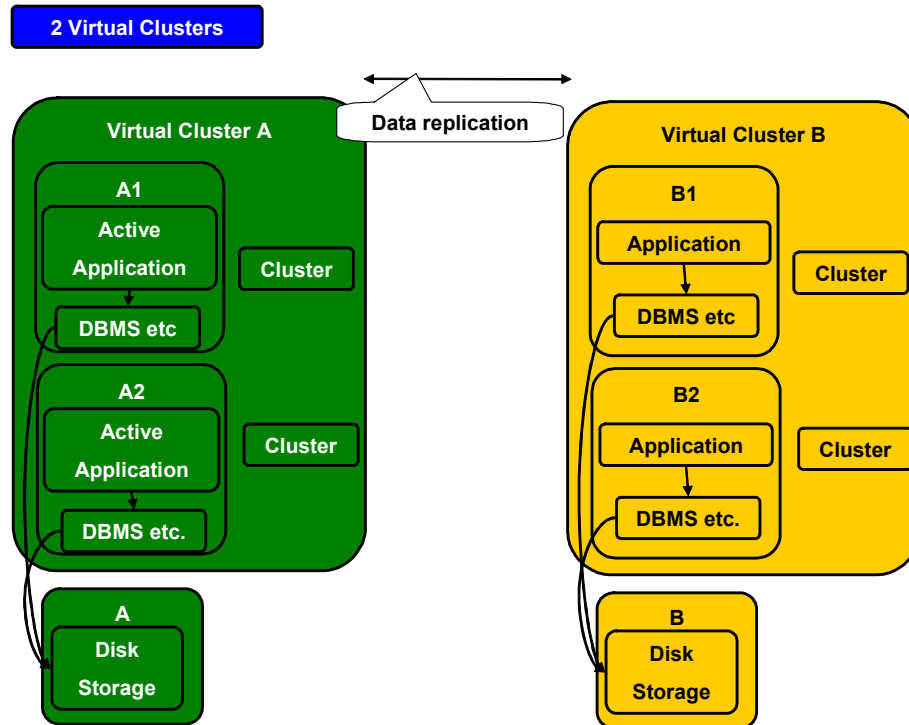


Figure 3-21 Two virtual clusters

2 Virtual Clusters

Sites	2
Active Sites	1..2
Servers	2
Storage Subsystems	1 + 1
Clustering	Y
Disk Mirror (n local,metro, regional, global)	regional or global or DB
DB Replication	Y or Disk Replication

Event ↓	Worst Case Action ↓	Impact on Service ↓
planned SW maintenance window	IPL OS Instance	No Outage
unplanned SW outage & restart	IPL OS Instance	No Outage-
planned HW maintenance window	Machine Restart	No Outage ; partial network switchover
server failure	Machine Repair & Restart	See Activity Level table
storage subsystem failure	Machine Repair & Restart	See Activity Level table
Building Failure	Building Repair	See Activity Level table
metro or network outage	BC DR Process	See Activity Level table

Figure 3-22 Features and availability characteristics of two virtual clusters

For the actions, RTO, and RPO for cases in the table in Figure 3-22 on page 37 where it says “See Activity Level table,” refer to Figure 3-14 on page 31.

Two systems

This scenario, shown in Figure 3-23, depicts two sites with no clustering at either site. Any planned or unplanned outage requires switching over completely to the other site, or being out of service. This is commonly seen in stable, medium size well-run environments that can bear a planned outage for a few hours every few months.

The features and availability characteristics of two systems are shown in Figure 3-24 on page 39.

Assuming that the application is running at Site A, Site B could also be active, handling a trickle, in ready state, warm, or cold.

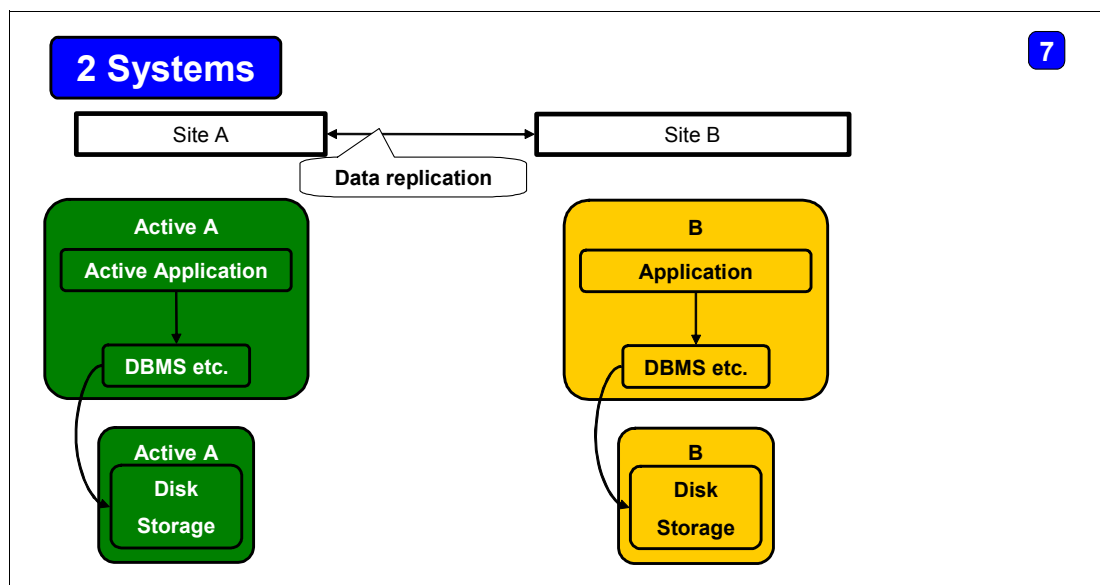


Figure 3-23 Two systems

2 Systems

7

Sites	2
Active Sites	1..2
Servers	2
Storage Subsystems	1 + 1
Clustering	N
Disk Mirror {n local,metro, regional, global}	regional or global or DB
DB Replication	Y or Disk Replication

Event ↓ V	Worst Case Action ↓ V	Impact on Service ↓ V
planned SW maintenance window	IPL OS Instance	No Outage ; partial network switchover
unplanned SW outage & restart	IPL OS Instance	See Activity Level table
planned HW maintenance window	Machine Restart	No Outage ; partial network switchover
server failure	Machine Repair & Restart	See Activity Level table
storage subsystem failure	Machine Repair & Restart	See Activity Level table
Building Failure	Building Repair	See Activity Level table
metro or network outage	BC DR Process	See Activity Level table

Figure 3-24 Features and availability characteristics of two systems

For the actions, RTO, and RPO for cases in the table in Figure 3-24 where it says “See Activity Level table,” refer to Figure 3-14 on page 31.

3.1.5 Summary

We have looked at the system structure within each site, considered activity levels and replication when there are two sites, and the overall two-site system structure.

A single-site physical cluster can provide continuous operation, except for a site outage or a loss of communications to the site.

This is summarized in Figure 3-25 on page 40.

One site

	1 instance	Virtual Cluster	Physical Cluster
Sites	1	1	1
Active Sites	1	1	1
Servers	1	1	2
Storage Subsystems	1	1	2
Clustering	-	Y	Y
Disk Mirror {n local,metro, regional, global}	n	n	local
DB Replication	-	-	-

Event ↓ V	Worst Case Action ↓ V	Impact on Service ↓ V	Impact on Service ↓ V	Impact on Service ↓ V
planned SW maintenance window	IPL OS Instance	down	No Outage	No Outage
unplanned SW outage & restart	IPL OS Instance	down	No Outage-	No Outage-
planned HW maintenance window	Machine Restart	down	down	No Outage
server failure	Machine Repair & Restart	down	down	No Outage-
storage subsystem failure	Machine Repair & Restart	down	down	No Outage- -
Building Failure	Building Repair	down	down	down
metro or network outage	BC DR Process	down	down	down

Figure 3-25 Single site

With data replication between sites, continuous availability can be achieved. Replication can be at the disk, database, or application level. Which levels and techniques are suitable varies depending on whether both sites operate with a single consistent set of information and how quickly one site must take over for the other.

This is summarized in Figure 3-26 on page 41.

2 Sites		2 Site Metro Dual Active Cluster + Replicated Storage	2 Site 1 Physical Cluster + 1 Virtual Cluster or Single System	2 Sites each with a Physical Cluster	2 Sites each with a Virtual Cluster	2 Sites each with one of everything
Sites		2	2	2	2	2
Active Sites		1..2	1..2	1..2	1..2	1..2
Servers		1..2 + 1..2	2 + 1	4	2	2
Storage Subsystems		1/site	2 + 1	2 + 2	1 + 1	1 + 1
Clustering		Y	Y	Y	Y	N
Disk Mirror		metro	local + regional or global or DB	local + regional or global or DB	regional or global or DB	regional or global or DB
DB Replication		-	Y or Disk Replication	Y or Disk Replication	Y or Disk Replication	Y or Disk Replication

Event ↓	Worst Case Action ↓	Impact on Service ↓	Impact on Service ↓	Impact on Service ↓	Impact on Service ↓	Impact on Service ↓
planned SW maintenance window	IPL OS Instance	No Outage ; partial network switchover	No Outage	No Outage	No Outage	No Outage ; partial network switchover
unplanned SW outage & restart	IPL OS Instance	No Outage- ; partial network switchover	No Outage-	No Outage-	No Outage-	See Activity Level table
planned HW maintenance window	Machine Restart	No Outage ; partial network switchover	No Outage	No Outage	No Outage ; partial network switchover	No Outage ; partial network switchover
server failure	Machine Repair & Restart	No Outage- ; partial network switchover	No Outage-	No Outage-	See Activity Level table	See Activity Level table
storage subsystem failure	Machine Repair & Restart	No Outage	No Outage- -	No Outage- -	See Activity Level table	See Activity Level table
Building Failure	Building Repair	No Outage- ; partial network switchover	See Activity Level table	See Activity Level table	See Activity Level table	See Activity Level table
metro or network outage	BC DR Process	No Outage- ; partial network switchover	See Activity Level table	See Activity Level table	See Activity Level table	See Activity Level table

Figure 3-26 Multiple sites

3.2 Factors that influence the approach to take on the target architecture

There are many determining factors that influence what is the most appropriate target architecture. In our experience these factors are not just technical but embrace elements that include business, process, cost and skills. In this section of the chapter we have outlined those that we think are most relevant. They are:

- ▶ Non-functional requirements (NFRs)
- ▶ Fit for purpose
- ▶ Cost
- ▶ Risk
- ▶ Business requirements
- ▶ Mapping source architecture to target architecture
- ▶ Common terminology

3.2.1 Non-functional requirements

Non-functional requirements (NFRs) are a key element in determining the qualities of service needed to support a specific application. Without a clear understanding of the NFRs such as volumes, *recovery time objectives (RTO)* and *recovery point objectives (RPO)* it is extremely

difficult to determine costs and time scales and may lead to scope creep later in the migration phase.

When migrating an application from the source platform, this application is likely to have been running on that platform for many years and the nonfunctional requirements were developed some time ago. Make sure you do not fall into the trap of assuming they stay the same. Revalidate them with the business and determine that metrics such as the RTO and RPO truly represent the needs of the business. Techniques to maintain the qualities of service will almost certainly differ from one platform to the next. Spend some time in any “design workshop” reviewing the NFRs and being very clear that you all agree on the outcome from this activity. This will have a significant effect on the cost of implementing the target architecture.

Capturing of NFRs is often an ad-hoc process, so when working on them try and use a more structured approach. Establish a methodology that defines the infrastructure characteristics of the target System z environment that you can use during the life cycle of the migration project and share with the wider team. The example in Figure 3-27 is a concept diagram that you can use to develop a spreadsheet with which you can capture the NFRs. This will be further developed in Chapter 7, “Transition approaches” on page 171.

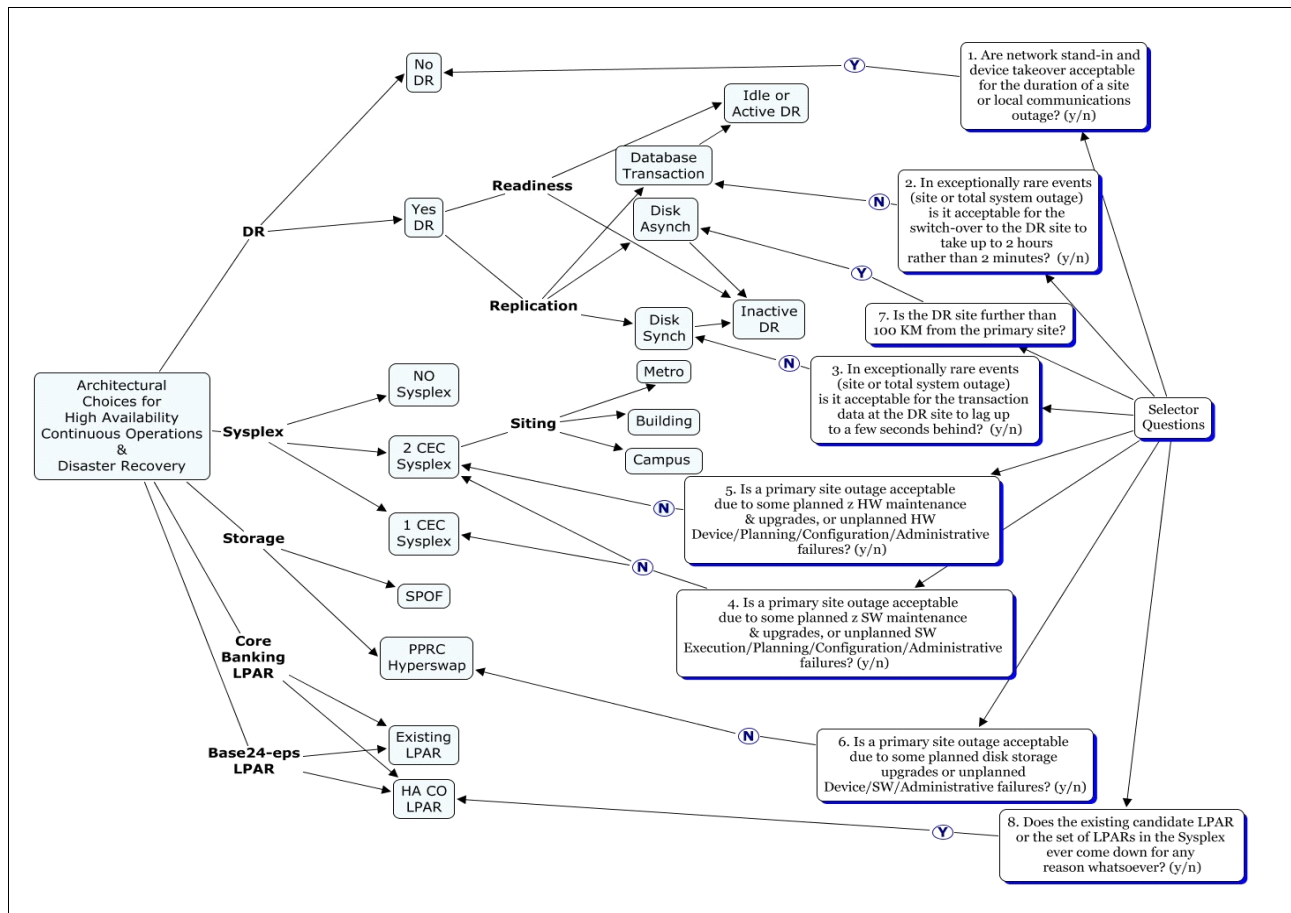


Figure 3-27 High availability architectural choices - concept diagram

3.2.2 Fit for purpose

In our experience people will naturally compare the source and target environments and determine capability in the context of the source environment. The architectures and

consequently the implementations of continuous availability and disaster recovery on different platforms will differ but the effect will often be the same. The System z environment in most cases is well established and the applications such as core systems will have a well established architecture that has been designed to meet the “qualities of service” (QoS) determined by the business functions they support. System z is a multipurpose environment designed to meet the needs of a diverse application set and has the flexibility to support different levels of availability and recovery based on need. In a distributed environment where the model is often “single application per server” this often does not apply.

When looking at the current QoS characteristics of a mainframe environment, the architecture and processes may not meet the needs of the new workload. In our experience the existing System z environments have been set up for some time. This may be because the importance of platform has grown or the workloads have grown and demand higher levels of availability. IBM will often work with clients to support them in enhancing the System z infrastructure in order to deliver higher availability, continuous operations, and continuous availability.

3.2.3 Cost of the target architecture

Cost will be a determining factor in terms of the target architecture. The more comprehensive it is in terms of quality of service capabilities the greater the cost. If the demands of the application are to deliver >99.999% availability, then there will be some configuration options that will increase costs significantly. An example could be the need to have the clustering technology duplexed. There are other components that may need to be duplexed in order to deliver the availability objectives.

When defining and comparing the costs of any migration you should be sure to include the following elements to ensure that you capture the complete list of associated items. They should cover:

- ▶ Hardware capital expenditure
- ▶ Hardware operating expense
- ▶ Software capital expenditure
- ▶ Software operating expense
- ▶ Data center capital expenditure
- ▶ Steady state labor costs
- ▶ Project labor costs
- ▶ Migration costs

When addressing the costs associated with running new applications on an existing or upgraded system, it is critical to understand its ability to run a mixed workload. We have discovered through analysis that often workloads will peak at different times. This consequently means that the capacity needed to drive a new workload can in part exist on the existing System z environment and only part of the capacity will need to be delivered through additional processor hardware.

If you also consider the flexibility in software pricing where it can be calculated based on a 4-hour rolling average, then you can further drive down the cost of application delivery.

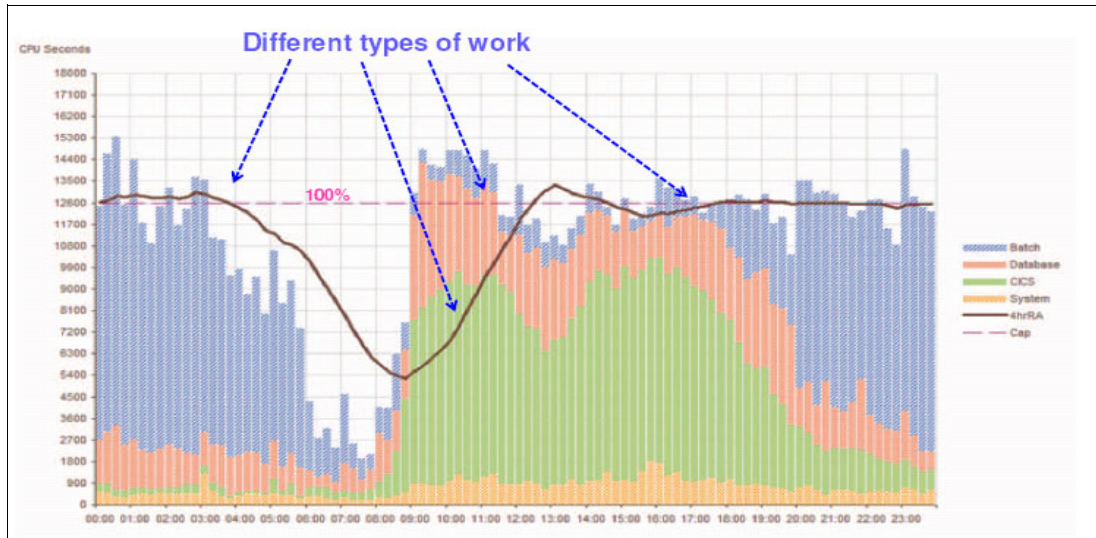


Figure 3-28 An example of a System z with multiple workloads

In addition to this there will be software cost benefits associated with shared capacity within a System z complex. This is not specific to high availability but a relevant part of the decision-making process.

3.2.4 Risk

Risk is an important focus for any platform migration project. It has to be evaluated in the context of both business and IT. Table 3-1 on page 44 shows a sample matrix that you can adopt when defining and quantifying project risk.

Ensure that the matrix framework effectively addresses all of the aspects of risk that are important for the project. Use it to identify and locate specific risk elements.

Define the specific risk elements and position them into this matrix as appropriate. For each risk element then define mitigation strategies where possible. Where possible identify risk valuation metrics.

Table 3-1 Risk assessment matrix - example

	Project risk	Operating risk
Technical risk	<ul style="list-style-type: none"> ▶ Stakeholders (key technical personnel) are committed ▶ Scope is realistic and managed ▶ Team is high performing ▶ Work and schedule are realistic 	<ul style="list-style-type: none"> ▶ Transition volumes far exceed original projections ▶ Additional drivers or interfaces are discovered during implementation ▶ Work and schedule are realistic

	Project risk	Operating risk
Migration risk	<ul style="list-style-type: none"> ▶ Stakeholders are committed ▶ Scope is realistic and managed ▶ Functional component migration estimates are realistic ▶ Team is high performing ▶ Delivery organization benefits are being realized 	NA
Business risk	<ul style="list-style-type: none"> ▶ Key change in business strategy creates conflict in terms of IT strategy and objectives ▶ Team is high performing ▶ Business benefits are realized ▶ Work and schedule are realistic 	<ul style="list-style-type: none"> ▶ Business benefits are impacted, for example pricing needs to be adjusted due to third party costs or other factors ▶ Team is high performing ▶ Work and schedule are realistic
Financial risk	<ul style="list-style-type: none"> ▶ Stakeholders are committed ▶ Risks are being clearly identified and mitigated 	<ul style="list-style-type: none"> ▶ Risks are mitigated ▶ Business benefits are impacted, for example pricing needs to be adjusted due to third party costs or other factors
Reputational risk	<ul style="list-style-type: none"> ▶ Stakeholders are committed ▶ Risks are being clearly identified and mitigated 	<ul style="list-style-type: none"> ▶ Disaster recovery and backup processes are appropriate to meet SLAs

3.2.5 Business requirements

The business requirements that underpin the service level agreement for the application are often dynamic and will change during the life of any application. When migrating, it is a perfect opportunity to go back to the business and revalidate the business requirements, which will further ensure that the cost case is viable. There is a term you often hear in IT: “We’ve always done it this way”. There should always be a well-defined process within IT service delivery that consistently validates the business case for all the applications that it serves. With the growing cost pressures on IT a migration project will always be seen as an additional cost unless IT can articulate the long term business benefits of achieving the target solution.

3.2.6 Mapping source architecture to target architecture

The “as-is” architecture from which you will derive the target architecture will have been designed on the basis of the capabilities of the application and underlying infrastructure. The high availability scenarios for System z are many and driven by the existing service levels. Isolating the specific needs of the application being migrated should be done in terms of both business requirements and capabilities of the platform. The service levels currently established on any existing System z environment may not be sufficient for the new workload but be fit for the purposes of the current applications running there. In order to deliver the service levels demanded to support the new high availability workload on System z certain changes may be needed to increase the capabilities of the infrastructure to meet the target

availability needs of the new workload. You can achieve this by isolating the workload on its own discreet Parallel Sysplex® that has all the key facilities for continuous availability and disaster recovery enabled, such as GDPS®, PPRC, and so on.

Some installations like to separate environments in that way but there are others that, because of well-established and tested processes and design approaches, feel comfortable deploying as many applications within the same sysplex as they can so that they can better exploit the capabilities that are available within a single Parallel Sysplex. There is no single right answer but there is the best answer for your environment.

3.2.7 Common terminology

Be sure that all team members working on the migration project have a common language and that there is no confusion on terminology. If not addressed, wrong assumptions will be made for the target architecture. At the beginning of this book we defined a set of commonly used terms in the context of continuous availability environments. An example is the term active/active, which can be interpreted differently.

Be sensitive to the different cultures that will be involved. People running distributed systems will not be familiar with System z best practices and will often be unaware of its capabilities to run many workloads concurrently within the same Parallel Sysplex environment.

Most distributed systems will run single applications on each server and would therefore have not been exposed to the multipurpose characteristics of System z, and therefore the associated terminology may not be understood.

3.3 Integration with external services

Few applications are an island. So you have to determine how you will handle the integration with external services when migrating to the target system. Examples may be credit checking, and card schemes for credit card authorization. The interfaces used in the source environment can often be old and unfit for the modern environment where integration is more sophisticated and driven by a more service-oriented approach.

Moving from a well-established distributed legacy application will pose some challenges for determining the best approach to take in the target environment. Moving to a message-based architecture may be a requirement as more SOA standards are adopted.

Older interfaces such as LU6.2 still exist in some installations. Having to set up a new network on the target System z environment may need additional architectural elements to support it. This may be the right opportunity to rationalize the disparate networking topologies that need to be supported. See 4.5, “Network and communications” on page 68.

An opportunity exists to transform to more up-to-date integration standards that are more SOA based. Decoupling of services that were imbedded within the source application will allow better sharing of systems and data resources. IBM recognizes the importance of this and has established frameworks that allow better deployment and integration of applications. There is a key decision to be made when migrating any application: “do I transform it” as part of the process. Of course it is important to determine to what extent the application needs to participate in a service-oriented environment. If that is the case then it would be a valid approach to take and re-engineer the application to exploit the SOA stack. In the past the application may have needed to use old fashioned approaches to interconnect to other applications and services. Decoupling as part of the transformation of the application may be a sensible strategic approach to take.

Older applications are probably static and own the data, and therefore moving to a new architecture provides opportunities to evaluate more relevant approaches to sharing of data that allow a more efficient data access approach.

Refer to 7.5, “Options for application modernization” on page 183 for more discussion about modernization and transformation of the application while migrating it to System z.



System z technology options

The System z platform is responsible for processing transactions for many of the world's applications, systems, and services across various industry sectors. For over 40 years, clients seeking the highest levels of availability have chosen System z as the platform to deliver mission-critical applications. The platform is designed for general purpose workloads and is capable of supporting web-based applications as well as traditional OLTP and batch applications.

In this chapter we provide a “snapshot” of the System z technology available for fulfilling the needs of mission-critical applications that need to be highly available. We address the following areas:

- ▶ "Operating systems" on page 50
- ▶ "Virtualization" on page 52
- ▶ "Resource management" on page 61
- ▶ "Workload management" on page 63
- ▶ "Network and communications" on page 68
- ▶ "Availability and scalability" on page 75
- ▶ "Security" on page 85
- ▶ "Transaction management" on page 96
- ▶ "Data management" on page 97
- ▶ "Programming environment" on page 107
- ▶ "Integration" on page 112
- ▶ "Systems management" on page 114
- ▶ "Specialty processors" on page 124

Note: If you are relatively unfamiliar with System z, this chapter is a good place to start to get an overview.

4.1 Operating systems

The System z platform supports various operating systems, with z/OS being the most widely used. The other operating systems supported are z/VM, Linux on System z, z/VSE™ and z/TPF.

Note: The latest generation of the System z platform, called zEnterprise, is capable of running applications on POWER7™ and x86 Blade extensions. However, in this book we focus on migrating highly available applications to Linux for System z and z/OS. Migration of applications on distributed platforms to zEnterprise Blade extensions is a separate discussion and requires a different approach, in our view.

4.1.1 z/OS

The z/OS operating system is the flagship operating system on System z, providing the highest levels of reliability, availability, maintainability, and security. The 64-bit operating system is designed for executing high volume, mixed workloads.

z/OS is structured around *address spaces*¹, which are ranges of addresses in virtual storage.

Note: The term *virtual storage* is used on z/OS as opposed to *virtual memory* on other platforms.

Each z/OS user gets an address space containing the same range of storage addresses. The use of address spaces in z/OS allows for isolation of private areas in different address spaces for system security, yet also allows for inter-address space sharing of programs and data through a common area accessible to every address space.

Technologies supported on z/OS include, but are not limited to, DB2®, CICS®, IMS™, WebSphere MQ, as well as C/C++, 64-bit Java and UNIX APIs and applications, including UNIX-style hierarchical file systems HFS and zFS.

A key technology for z/OS is *Parallel Sysplex*, which allows multiple z/OS partitions, on the same machine or different machines, to be clustered and operate as a single *logical server*. A Parallel Sysplex can consist of 2 to 32 systems on any mix of System z servers that support the technology. In a Parallel Sysplex environment, it is possible that the loss of a server may be transparent to the application, and the server workload can be redistributed automatically within the Parallel Sysplex with little or no performance degradation. Therefore, events that otherwise would seriously impact application availability, such as failures in the *Central Processor Complex (CPC)* hardware elements or critical operating system components, would, in a Parallel Sysplex environment, have reduced impact. *Workload Manager (WLM)* provides the Parallel Sysplex cluster with the intelligence to determine where work needs to be processed and in what priority. The priority is based on the client's business goals and is managed by Parallel Sysplex technology.

An extension of Parallel Sysplex is *Geographically Dispersed Parallel Sysplex™ (GDPS)* which is the primary disaster recovery and continuous availability solution for a multisite enterprise. GDPS automatically mirrors critical data and efficiently balances workload between the sites.

¹ More information on address spaces in "Address spaces in z/OS" on page 59

4.1.2 z/VM

z/VM is an operating system for the IBM System z platform that provides a highly flexible test and production environment. The z/VM implementation of IBM virtualization technology provides the capability to run full-function operating systems such as Linux for System z, z/OS, z/VSE and z/TPF as *guests*. z/VM supports 64-bit IBM z/Architecture® guests and 31-bit IBM Enterprise Systems Architecture/390 guests. For more information on running other operating systems as guest under z/VM, refer to *z/VM: Running Guest Operating Systems*, SC24-6115-03.

z/VM provides each user with an individual working environment known as a *virtual machine*. The virtual machine simulates the existence of a dedicated real machine, including processor functions, memory, networking, and input/output (I/O) resources. Operating systems and application programs can run in virtual machines as guests. For example, you can run multiple Linux and z/OS images under the same z/VM system that is also supporting various applications and users. As a result, development, testing, and production environments can share a single physical computer. A virtual machine uses real hardware resources, but even with dedicated devices (such as a tape drive), the virtual address of the tape drive may or may not be the same as the real address. Therefore, a virtual machine only knows “virtual hardware” that may or may not exist in the real world.

There is more discussion on z/VM, from a virtualization perspective, in “z/VM” on page 55.

4.1.3 Linux on System z

Linux on System z is available via Novell SUSE or Red Hat distribution and can run in native LPAR mode or as a guest under z/VM. Linux on System z is an attractive proposition for enterprises interested in reducing costs by consolidating servers. Typically, in a Linux environment, each single application might run on a physical server. The networking demands and complexity of running these systems can be greatly reduced by transferring them to run as guests under z/VM. It is possible to run hundreds of Linux servers on one physical machine. Here we list some of the advantages of running Linux under z/VM:

- ▶ Sharing of resources:
 - Processor
 - I/O
 - Storage (using Linux as an NSS)
- ▶ Maintainability (multiple guests sharing the same kernel and data)
- ▶ Reliability
- ▶ Exploitation of the unique features of z/Architecture
- ▶ Centralized management
- ▶ Greater security using cryptographic coprocessors
- ▶ Cost savings on software licensing
- ▶ Less energy and cooling required

There is more discussion on Linux on System z in “Ten good reasons to run Linux as a guest of z/VM” on page 58.

4.1.4 Important decisions with respect to operating systems

With respect to the choice of the operating system on System z, the following decisions have to be made when adding a new workload to the environment:

- ▶ Does the workload require z/OS qualities of service?
- ▶ If Linux is the best fit, is there a need for a large number of Linux systems?
If this is the case, then running Linux as guest under z/VM is the way to go.
- ▶ In case of Linux, which distribution is needed (Red Hat or Novell SUSE)?
- ▶ Are one or multiple Parallel Sysplexes needed?
- ▶ Are there specific requirements that could warrant investigating the usage of z/VSE or z/TPF?

Note: Due to their very specific nature we have left z/VSE and z/TPF outside our consideration. You may, however, benefit from these operating systems in specific situations.

4.2 Virtualization

The System z architecture was designed with the concept of *sharing*. Sharing starts in the hardware components and ends with the data that is being used by the platform. The ability to share everything is based on one of the major strengths 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. Virtualizing the System z environment involves creating virtual systems (logical partitions and virtual machines), and assigning virtual resources (such as processors, memory, and I/O channels) to them. Resources can be dynamically added or removed from these logical partitions through operator commands.

4.2.1 Mainframe virtualization technologies

The virtualization capabilities of the IBM mainframe represent some of the most mature and sophisticated virtualization technologies in the industry today. For example, a single IBM System z mainframe can scale up to millions of transactions per day or scale out to manage tens to hundreds of virtual servers. It can also redistribute system resources dynamically to manage varying server demands on the system resources automatically.

The following are key technologies that enable these virtualization capabilities:

- ▶ At the System z level:
 - PR/SM™ and logical partitioning
 - z/VM
- ▶ At the z/OS level:
 - Address spaces
 - Enclaves

We discuss these topics in more detail in the following sections.

4.2.2 PR/SM and logical partitioning

In a server environment with distributed midrange processors, it is typical to find a single (or no more than a few) application housed or supported by an entire physical server. On an IBM System z processor, it is common to find many applications and workloads running on the same physical server, under any of the supported operating systems. A System z server can be carved up into multiple logical partitions (LPARs) with a mixture of Linux for System z and z/OS, all Linux for System z or all z/OS LPARs. An LPAR on System z is managed as a logical system, which means it behaves as if it were a physical server.

Important: The concept of LPARs on System z provides safe coexistence between applications and application environments.

LPARs and LPAR management on System z

Processor Resource/System Manager (PR/SM) is a *Type 1 hypervisor* integrated with all IBM System z processors that allows physical resources to be carved up so that many logical partitions (LPARs) can be created and share the same physical resources. Each logical partition operates independently, running its own operating environment.

Note: A Type 1 hypervisor runs directly on the hardware.

LPARs can be configured according to the type of work they are to support. A System z server, for example, can house LPARs to support development, test, quality assurance, production or even a sandbox systems programmer environment. As indicated, LPARs can also be configured across multiple machines in a Parallel Sysplex environment in one or multiple sites. Typically, in a Parallel Sysplex environment LPARs are defined in pairs across machines by work type (development, quality control, or production) or applications that they support. This provides redundancy for high availability.

In any case, separating work by LPAR by type provides another level of safe coexistence. By separating development from production allows the less stable test work to be segregated from the production environment since the development work is subject to abends and looping code that can consume significant resources. Processor resources for mixed workloads are better managed and controlled when separated into LPARs by workload type (production vs. development).

LPAR management involves use of LPAR definition parameters entered into the System z *Hardware Management Console (HMC)* by the configuration specialist in order to allocate resources among the LPARs defined on the System z server. These parameters include:

- ▶ Number of *control processors (CPs)* or *engines*
- ▶ Percentage of processor capacity (weight)
- ▶ Amount of central storage memory
- ▶ Use of cryptographic coprocessors
- ▶ Number of *specialty engines* and their weight²

Figure 4-1 on page 54 illustrates this concept of carving up a physical server into LPARs, with different operating systems and for different purposes. A few examples with respect to this diagram are:

- ▶ LPARs 1 through 4 all run dedicated z/OS operating systems for production and preproduction purposes.

² Refer to "Specialty processors" on page 124 for more details.

- ▶ LPAR 5 runs z/VM with both z/OS and Linux guests.
- ▶ LPARs 6 and 7 run a dedicated Linux system.
- ▶ LPAR 8, again, runs z/VM, but with only Linux guests.

All these LPARs ultimately consume CPs that are part of the hardware. As you can read later in this chapter, there are different types of CPs on System z, which in fact are all physically the same CPs, but are different in terms of cost and usage.

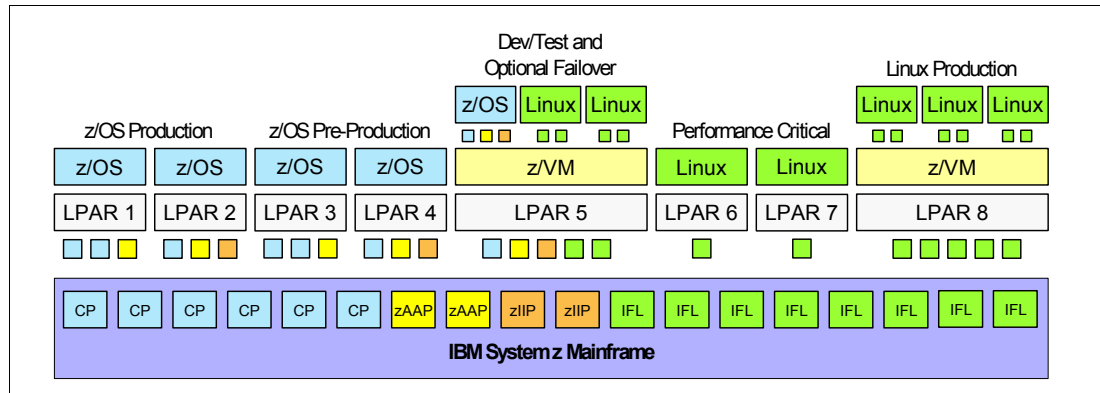


Figure 4-1 The power and flexibility of System z virtualization

From a weight perspective, each LPAR is guaranteed to get the percentage of the processor defined for it and can only steal cycles (given that CPs are shared among LPARs) if other LPARs are not using their weight. Therefore, a development LPAR that may require more cycles than its defined weight can only access those cycles if production LPARs on the same machine do not need them. For example, a development LPAR may require more cycles to process peak workloads or as a result of a looping task. This defining of processor cycles provides built-in protection of workloads and does not allow one LPAR to dominate a machine at the expense of other LPARs unless the other LPARs do not need their defined cycles. This provides for safe coexistence of mixed workloads on the same System z server from a processor resource allocation perspective.

Of course, there is another scenario, whereby one or more CPs on a System z server can be dedicated to an LPAR (as opposed to being shared among LPARs). This would provide resource capping for an LPAR and its supported workloads. Effectively, it would also provide a level of resource protection against looping programs and unpredicted workload growth in the LPAR from grabbing too much of the total processor capacity in the box.

Memory must be configured in fixed increments on each LPAR depending on the total physical memory existing on the System z server. This granularity of memory that must be reflected in the LPAR definition parameters is shown in Table 4-1 on page 55.

Table 4-1 LPAR memory increments

increment	Total processor storage
16 MB	Less than 8 GB
32 MB	More than 8 GB but less than 16 GB
64 MB	More than 16 GB but less than 32 GB
128 MB	More than 32 GB but less than 64 GB
256 MB	More than 64 GB but less than 128 GB
512 MB	More than 128 GB

LPAR management also allows for reserving resources on an LPAR basis to satisfy legitimate emergency requirements or unusual peak resource demands. For example, additional CPs and memory can be designated within the HMC for each LPAR via the LPAR definition parameters. CPs designated for reserve could be currently shared by other LPARs or not physically used by any LPAR (currently in offline mode) while reserve storage would be physical storage not used by any LPAR on the System z processor. Through z/OS console commands, the extra CPs or memory previously designated in reserve could be varied on to a given LPAR with no impact to other LPARs on the same machine (safe coexistence).

In addition, the processor weights of LPARs on a System z machine can be dynamically altered via the HMC to redistribute processor resources. Essentially, the guaranteed minimum cycles could be increased on one LPAR while decreased for other LPARs on the same System z machine. On a processor that is running close to 100%, however, it is conceivable that decreasing weights on an LPAR could have a negative impact on performance for workloads on that LPAR, but it is assumed that it will be a management decision to steal cycles from lower priority work (probably a development LPAR) to enhance performance for a production LPAR.

In terms of cryptography, the definition of crypto parameters in the HMC for an LPAR allows for the assignment of crypto coprocessors to a given LPAR. A unique master key is then assigned by the MVS systems programmer, which then allows each database (IMS, DB2, and so on) on the LPAR to be loaded via a load utility or API using crypto technology (such as Clear Key) and then subsequently accessed via crypto database (IMS, DB2, and so on) access modules. Use of crypto technology provides protection from unintentional or unauthorized access of database data between LPARs or within the same LPAR or from an outside source.

4.2.3 z/VM

In 4.1.2, “z/VM” on page 51 we talked briefly about z/VM as one of the available operating systems on System z. Besides its operating system functionality, z/VM also fulfills a role as virtualization layer. It provides each user with an individual working environment known as a *virtual machine (VM)*. The virtual machine uses virtualization to simulate the existence of a real machine by sharing resources of a real machine, which include processors, storage, memory, and input/output (I/O) resources.

Operating systems and application programs can run in virtual machines as guests. For example, you can run multiple Linux and z/OS images on the same z/VM system that is also supporting various applications and users. As a result, development, testing, and production environments can share a single physical computer.

Figure 4-2 on page 56 shows an example of a configuration of an LPAR with z/VM. A first-level z/VM means that it is the base operating system that is installed directly on top of the real hardware. A second-level system is a user brought up in z/VM where an operating system can be executed on the first-level z/VM.

In other words, a first-level z/VM operating system sits directly on the hardware, but the guests of this first-level z/VM system are virtualized. By virtualizing the hardware from the first level, you can create and use as many guests as needed with a small amount of actual real hardware.

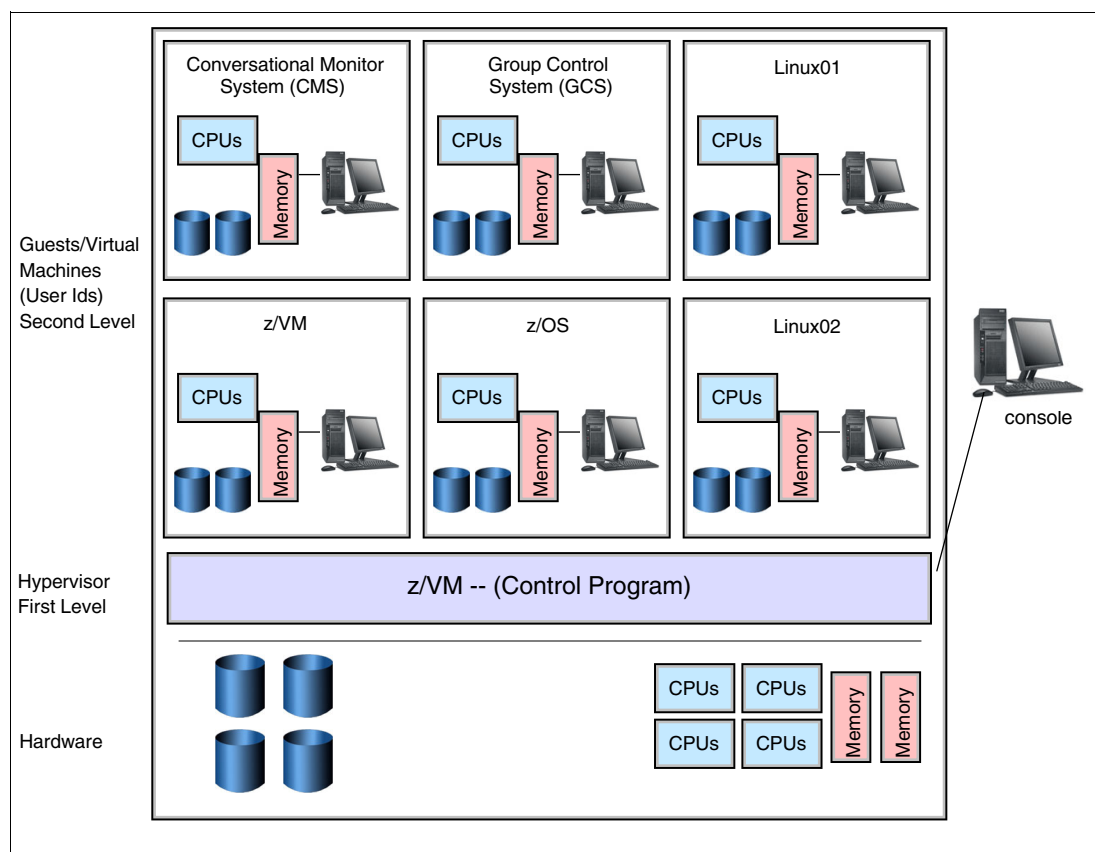


Figure 4-2 Sample configuration of an LPAR

z/VM consists of many components and facilities that bring the reliability, availability, scalability, security, and serviceability characteristics of System z servers, such as:

- TCP/IP for z/VM brings the power and resources of your mainframe server to the Internet. Using the TCP/IP protocol suite of TCP/IP for z/VM, you can reach multiple vendor networking environments from your z/VM system. TCP/IP for z/VM enables z/VM systems to act as peers of other central computers in TCP/IP open networks. Applications can be shared transparently across z/VM, Linux, and other environments. Users can send messages, transfer files, share printers, and access remote resources with a broad range of systems from multiple vendors.

More information on TCP/IP for z/VM can be found in "The z/VM TCP/IP stack" on page 72.

- *Open Systems Adapter-Express (OSA-Express)* and *Open Systems Adapter Express2 (OSA-Express2)* are integrated hardware features that enable the System z platform to provide industry-standard connectivity directly to clients on local area networks (LANs) and wide area networks (WANs).

- The *Resource Access Control Facility (RACF®)* Security Server for z/VM is a security tool that works together with existing functions in the z/VM base system to provide improved data security for an installation. RACF protects information by controlling access to it. RACF also controls what you can do on the operating system and protects your resources. It provides this security by identifying and verifying users, authorizing users to access protected resources, and recording and reporting access attempts.

Contrasted with a discrete server implementation, z/VM-based System z solutions are designed to provide significant savings, which can help lower total cost of ownership (TCO) for deploying new business and enterprise application workloads on a mainframe.

How z/VM virtualization works

The heart of z/VM is a multiprogramming, multiprocessing operating system kernel known as the *Control Program (CP)*. The CP is the component of z/VM that creates and dispatches virtual machines on the real System z hardware.

The CP supports hundreds of commands, including the configuration of the virtual machine, and it lets users change virtual machine configurations nearly at will.

z/VM virtualization covers processors, memory, and I/O devices.

- Virtualization of processors

Because the z/Architecture defines that a System z data processing system can house 1 to 64 processors, each virtual machine z/VM creates can have 1 to 64 virtual processors. z/VM provides control over processor resources by letting a system administrator assign a share value to each virtual machine.

z/VM also lets the system administrator define a maximum share value to prevent a guest from excessively consuming processor resource. The z/VM system administrator or system operator can adjust share settings while virtual machines are running.

- Virtualization of memory

z/VM lets virtual machines share memory, which helps reduce memory requirements. All guest memory is virtual. CP overcommits physical memory by keeping resident only those guest pages that appear to have been needed in the recent past. When physical memory is scarce, CP moves stagnant guest pages first to expanded storage (a high-speed page storage buffer) and eventually to disk. CP brings these pages back to memory if the guest ever needs them again.

- Virtualization of I/O devices

z/VM uses various methods to provide devices to virtual machines. CP can dedicate, or attach, a real device to a virtual machine. This gives the virtual machine exclusive use of the entire real device. CP can also virtualize a device, which means it gives a guest a portion of a real device. This can be a portion in time, such as of a processor, or a portion of the device's storage capacity, such as of a disk drive.

Network connectivity is an important concern in many environments. z/VM meets clients' network needs by offering several networking options. The Control Program can dedicate network devices to virtual machines. The dedicated device can be a channel-to-channel adapter, an IBM Open Systems Adapter (OSA) that provides Ethernet connectivity, or a HiperSockets™ device, a kind of network adapter that connects one LPAR to another. z/VM also has its own TCP/IP stack, which guests can use as though it were an IP router.

A common network option used today is the *virtual switch*. Here, CP equips each virtual machine with a simulated IBM OSA and connects all those simulated OSAs to a simulated LAN segment called a guest LAN. Also connected to the guest LAN is a real OSA that CP manages. With this configuration established, CP can provide packet- or frame-switching

functions for the guests, just as a real switch would in a real external network. In this way, the guest LAN becomes an extension of a real external LAN segment.

Ten good reasons to run Linux as a guest of z/VM

Running the Linux operating system as a guest of z/VM is a smart choice. Consider the following benefits z/VM offers a Linux guest environment. Also see:

<http://www.vm.ibm.com/linux/benefits.html>

- ▶ **Sharing resources**
Resources can be shared among multiple Linux images running on the same z/VM system. These resources include processor cycles, memory, storage devices, and network adapters.
- ▶ **Server hardware consolidation**
Running tens or hundreds of Linux instances on a single System z server offers clients savings in space and personnel required to manage real hardware.
- ▶ **Virtualization**
The virtual machine environment is highly flexible and adaptable. New Linux guests can be added to a z/VM system quickly and easily without requiring dedicated resources. This is useful for replicating servers in addition to giving users a highly flexible test environment.
- ▶ **System z advantages**
Running Linux on z/VM means the Linux guest or guests can transparently take advantage of z/VM support for System z hardware architecture and Reliability, Availability, and Serviceability (RAS) features.
- ▶ **z/VM connectivity**
Using guest LANs and virtual switches (VSWITCH), z/VM provides high-performance communication among virtual machines running Linux and other operating systems on the same processor. Simplification of the network by using HiperSockets can provide savings and reduce cabling, hubs, switches, and routers. It can also help to reduce the maintenance effort.
- ▶ **Minidisk driver**
Linux on System z includes a minidisk device driver that can access all DASD types supported by z/VM.
- ▶ **Data-in-memory**
Data-in-memory performance boosts are offered by VM exploitation of the z/Architecture.
- ▶ **Debugging**
VM offers a rich debug environment that is particularly valuable for diagnosing problems in the Linux kernel and device drivers.
- ▶ **Control and automation**
The long-standing z/VM support for scheduling, automation, performance monitoring and reporting, and virtual machine management is available for Linux virtual machines.
- ▶ **Horizontal growth**
An effective way to grow your Linux workload capacity is to add more Linux guests to a z/VM system.

4.2.4 Address spaces in z/OS

System/370 was the first IBM architecture to use virtual storage and address space concepts emulating the first virtual system. The address space is a set of contiguous virtual addresses available to a program's instructions and its data. The range of virtual addresses available to a program starts at 0 and can go to the highest address permitted by the operating system architecture. The address space size is decided by the length of the fields that keep such addresses. Because it maps all of the available addresses, an address space includes system code and data as well as user code and data. Thus, not all of the mapped addresses are available for user code and data. The S/370 architecture used 24 bits for addressing, so the highest accessible address in the MVS/370 was 16 megabytes, which was also the address space size.

Figure 4-3 depicts the address space concept.

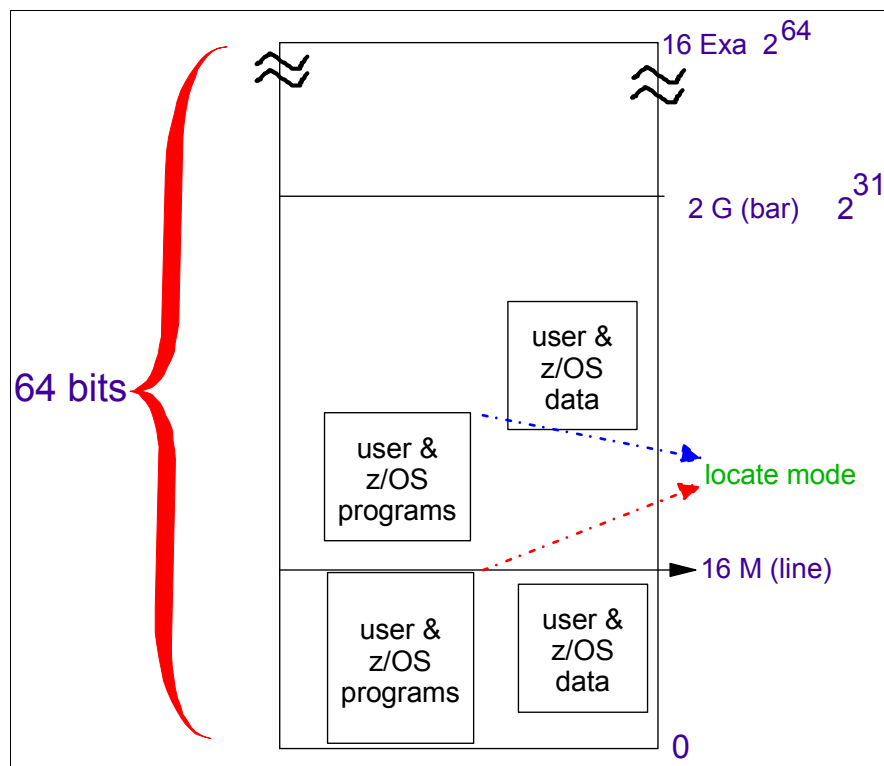


Figure 4-3 Address space concept

Over time programs became more complex and there was a requirement to support larger and larger programs. This requirement fueled the need to support larger address spaces with enhanced addressability schemes.

With MVS/XA, the XA architecture extended to 31 bits for addressing and the address space size went from 16 MB to 2 GB. The 16 MB address became the division point between the two architectures and is commonly called the *line*.

For compatibility, programs running in MVS/370 should run in MVS/XA, and new programs should be able to exploit the new technology. The high-order bit of the address (4 bytes) is *not* used for addressing, but rather to indicate to the hardware how many bits are used to solve an address: 31 bits (bit 32 on) or 24 bits (bit 32 off).

The z/Architecture extended to 64 bits with z/OS and the address space size went from 2 GB to 16 exabytes. The 2 GB address is called the *bar*. The addresses above the bar are used for data only. For compatibility, programs running in MVS/370 and MVS/XA should run in z/Architecture.

There are several types of address spaces running under z/OS, each supporting a different type of work. These address space types include (but are not limited to):

- ▶ Started tasks (STCs)
- ▶ TSO sessions
- ▶ Batch
- ▶ Unix Systems Services (USS) or OMVS (for supporting UNIX processes)
- ▶ Distributed Data Facility (DDF) for DB2

Addressing mode and residence mode

With MVS/XA came the concept of *addressing mode* (AMODE); see Figure 4-4. AMODE is a program attribute to indicate which hardware addressing mode should be active to solve an address (that is, how many bits should be used for solving and dealing with addresses).

- ▶ AMODE=24 indicates that the program may address up to 16 MB virtual addresses.
- ▶ AMODE=31 indicates that the program may address up to 2 GB virtual addresses.
- ▶ AMODE=64 indicates that the program may address up to 16 exabyte virtual addresses (only in z/Architecture).

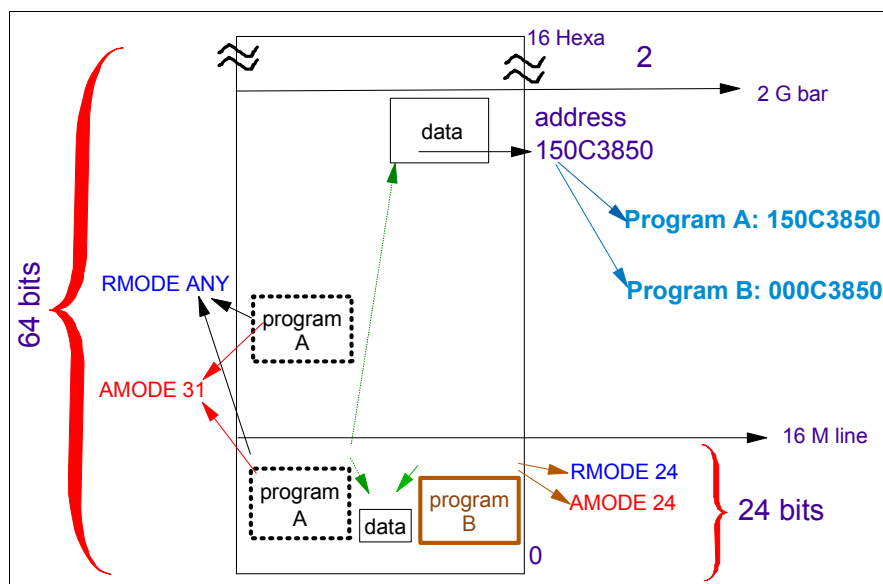


Figure 4-4 Addressing mode and residence mode

The concept of *residence mode* (RMODE) is used to indicate where a program should be placed in the virtual storage (by z/OS program management) when the system loads it from DASD, as explained here:

- ▶ RMODE=24 indicates that the module *must* reside below the 16 MB virtual storage line. Among the reasons for RMODE24 is that the program is AMODE24. The program has control blocks that must reside below the line.
- ▶ RMODE= ANY indicates that the module may reside anywhere in virtual storage, but preferably above the 16 MB virtual storage line. Because of this, such an RMODE is also

called RMODE 31. Note that in z/OS there is no RMODE=64 because the virtual storage above 2 GB is not suitable for programs, only data.

AMODE and RMODE are load module attributes assigned when load modules are created by the Binder program and are placed in a load module's directory entry in a partitioned data set.

4.2.5 Enclaves on z/OS

Workload Manager manages three kinds of z/OS workloads: address spaces, enclaves, and CICS and IMS transactions, which fall into a category by themselves. Most people readily understand z/OS address spaces, but what about enclaves? They're a type of z/OS work where transactions are managed independently of the address space in which they're running. Transactions that run in enclaves let WLM assign them a goal that results in a dispatching priority separate from the address space's goal and dispatching priority.

Task Control Block (TCB) work and preemptible *Service Request Block (SRB)* work (called enclave SRBs) are scheduled into an enclave and associated with its goal. These enclave SRBs are pieces of work z/OS can separately dispatch. Distributed DB2 transactions, DB2 stored procedures, and WebSphere Application Server transactions all use enclave SRBs; CICS and IMS subsystems do not use enclaves. CICS and IMS transactions are not pieces of work that z/OS can separately dispatch.

Enclaves were developed and became necessary to handle the complex internal flows of some new transaction managers, such as:

- ▶ DB2 Distributed Data Facility (DDF) V4 and later
- ▶ DB2 Stored Procedures in V5 and later
- ▶ IBM HTTP Server for OS/390®
- ▶ MQSeries® Workflow

4.2.6 Important decisions with respect to virtualization

With respect to virtualization on System z, the following decisions have to be made when adding a new workload to the environment:

- ▶ How many partitions (LPARs) are needed in total?
- ▶ How many LPARs are needed for dedicated z/OS or Linux systems?
- ▶ Can you exploit the benefits of running Linux or z/OS as a guest under z/VM?
- ▶ How will resources be assigned to the various LPARs on the system?
- ▶ On z/OS, how many address spaces (regions) will you finally need for the workload?

4.3 Resource management

We now discuss two key technologies on System z for resource management:

- ▶ Intelligent Resource Director (IRD)
- ▶ HiperDispatch

Intelligent Resource Director

Intelligent Resource Director (IRD) is a feature that extends the concept of goal-oriented resource management by allowing to group system images that are resident on the same

System z running in LPAR mode, and in the same Parallel Sysplex, into an LPAR cluster. This gives Workload Manager the ability to manage resources, both processor and I/O, not just in one single image, but across the entire cluster of system images.

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 (processor and channels) to the LPAR where it is most needed.

IRD addresses three separate but mutually supportive functions:

- ▶ LPAR processor 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 processor resources that are allocated to a logical partition when there is competition for processor resources 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 processor capacity they require. If the workload mix changes, the weight must be reevaluated and manually changed.

The goal of LPAR processor management is to help simplify the configuration task by automatically managing physical processor resources to allow high utilization of physical processor capacity, while allowing performance objectives to be met at times of peak demands. IRD LPAR processor management extends WLM goal-oriented resource management to allow for dynamic adjustment of logical partition processor weight. This function moves processor capacity 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.

- ▶ Channel Subsystem Priority Queuing

This new function is used 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 Queuing 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 Queuing.

- ▶ 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 Queuing.

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 do not 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).

HiperDispatch

HiperDispatch is a function that combines the dispatcher actions and the knowledge that PR/SM has about the topology of the server. HiperDispatch manages the number of logical

CPs in use. It adjusts the number of logical processors within a logical partition to achieve the optimal balance between CP resources and the requirements of the workload in the logical partition in cooperation with the weight management part of LPAR processor management.

HiperDispatch also adjusts the number of logical processors. The goal is to map the logical processor to as few physical processors as possible. Doing this efficiently uses the CP resources by attempting to stay within the local cache structure, making efficient use of the advantages of the high-frequency microprocessors and improving throughput and response times.

4.4 Workload management

In the following sections we discuss workload management in z/OS and Linux on System z.

4.4.1 Workload management under z/OS

Work enters a z/OS image in many ways. A task could be started (a started task) via a console command or an automation package. Transactions arrive by users connected to the System z processor and are managed by transaction management address spaces (CICS, IMS, and so on) or TSO sessions. *Distributed Data Facility (DDF)* transactions could come in from distributed users connected remotely to the System z server submitting SQL queries to access DB2 databases.

Work could also arrive (be submitted) as batch in a z/OS image, through TSO, a job scheduling package, other environments or platforms, or other LPARs. This batch is controlled under z/OS by means of *initiators* which can be managed by the *Job Entry Subsystem (JES)* or Workload Manager (WLM). In either case the initiators are grouped by a *class* to allow a mixture of different workload types and use of available cycles.

Under z/OS, it is the function of WLM to manage mixed workloads on an LPAR or across LPARs in a sysplex. The WLM policy classifies work into service classes through a set of classification rules. Attached to each service class is an importance or prioritization in the sysplex. When the importance is defined correctly, it ensures quality of service and safe coexistence of workloads.

The importance level values can be:

- ▶ Highest (1)
- ▶ High (2)
- ▶ Medium (3)
- ▶ Low (4)
- ▶ Lowest (5)

The absolute value specified for a service class is meaningless. What matters is the *relative* value.

A single WLM policy is defined, installed and activated sysplex wide and typically would give a higher importance to online (TSO, CICS, and so on) work versus batch. In addition to importance, WLM defines a business goal to each service class, which designates a desired turnaround for the type of work to be processed in the service class. Different types of goals can be specified, such as response time goals for online, transaction, or enclave type work versus velocity (speed) goals for batch. Response time goals can be expressed in terms of average or percentile of meeting a goal. Figure 4-5 on page 64 shows examples of goals.

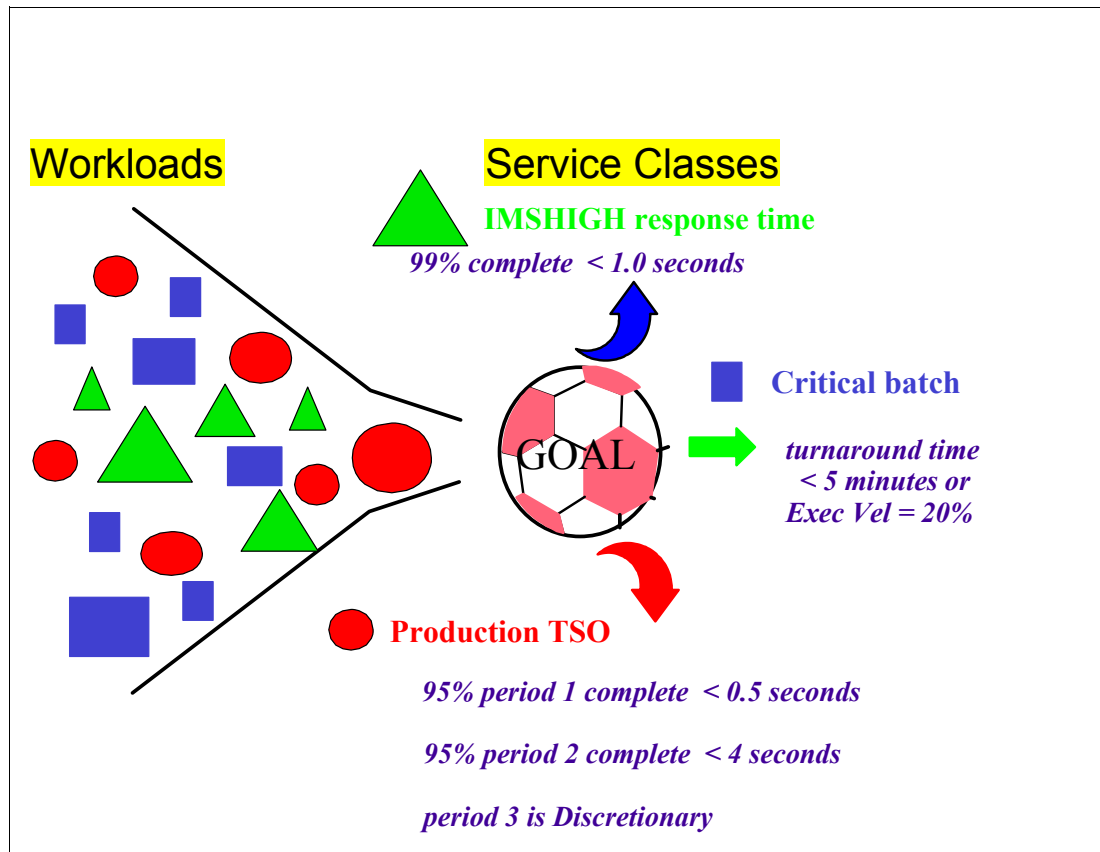


Figure 4-5 WLM goals

Important: In any case, WLM controls and distributes resources to the address spaces and enclaves based on the importance of the service class to which they have been assigned in order to achieve the business goals stated in the WLM policy.

Typically, for transactions such as CICS and IMS, goals are specified at the region level or optionally at a more granular transaction level (but only for the most important transactions in order to minimize overhead of the WLM policy). In any case, WLM then allocates resources to the address spaces serving the transactions (CICS or IMS regions) in order to achieve goals. If there are multiple transactions defined with goals that run in the same region, then WLM would assign importance and resources to the address space housing the transactions based on the transaction with the most stringent goals. However, the less important transactions also running in the same CICS regions will benefit and will go along for a “free ride”, possibly getting better service than they need.

In terms of WLM enclave processing, each individual enclave in a server address space is assigned resources based on the WLM goals. The less important enclave transactions therefore do not get a free ride when executing in the same address spaces as more important enclave work. Therefore, with WLM enclave processing, it is possible to consolidate diverse work on a set of address spaces and have the most important transactions meet their goals and not be impacted by less important transactions running in the same set of address spaces.

It is important to remember that when new work (address space, transaction, or enclave) is introduced to an existing LPAR running under z/OS, it must be covered by a definition in the WLM policy. Otherwise, it will receive the default “bottom of the barrel” priority and in many

cases (especially on a fully loaded z processor) suffer performance degradation. The type of definition required for new work in WLM will, of course, depend on granularity of existing workload definitions in the WLM policy. Since WLM policy definitions are effective sysplex wide, the same principle regarding WLM definitions also holds true for adding a new LPAR supporting new work in an existing plex.

Workload protection

Programs within jobs or tasks are subject to going into a loop. There are mechanisms within z/OS that can minimize the impact and provide protection against loops. First, by promoting work from a sandbox or development environment to a quality and/or production environment, you will minimize the occurrence of loop situations by shaking out the code before it reaches production.

Other mechanisms to provide loop protection include (but are not limited to):

- ▶ Resource capping

As mentioned above, an LPAR can be capped for processor resources by having a dedicated CP (or group of CPs) assigned to it.

- ▶ Resource groups

A resource group can be defined in WLM with a processor cycle minimum as well as a maximum (cap) per unit of time. This ensures that a task or group of tasks will not monopolize the processor resources allocated to an LPAR.

- ▶ DB2 governor

The governor oversees the processing of a DB2 transaction and can cap the processor activity on a transaction basis.

- ▶ Monitoring and alerting

Through performance monitoring and/or alerting, loops can be detected utilizing tools such as the Tivoli® Omegamon XE product suite.

Besides loop protection, there are various other mechanisms to provide protection to workloads in an LPAR or across LPARs in terms of access to resources. These include (but are not limited to):

- ▶ Storage protection

Programs can be coded to access and/or allocate storage (memory) that is key-protected (utilizing the key 0 to 8 protection scheme).

- ▶ Security

Security packages such as RACF allow security rules to be written to restrict access to facilities (such as TSO or CICS regions) and resources (such as DASD or tape files and databases).

- ▶ Global resource serialization (GRS)

This facility provides serial access to a given file or resource name to provide data integrity across LPARs or within the same LPAR.

- ▶ Crypto

As indicated above, crypto provides a lower level of data integrity protection against the unintentional or unauthorized access of data stored on a media such as tape or DASD. Unless authorized to unencrypt the data through the crypto access method modules (DB2, and so on) or APIs, access to data is likely to be unusable.

- ▶ EAL5 certification

The EAL 1 to 7 designations reflect a numerical grade assigned to an application after a security evaluation utilizing common criteria from an international standard in effect since 1999. The security evaluation examines design documentation, design analysis, functional testing and penetration testing. The EAL5 designation reflects the level of confidence that security features are reliably implemented for the application at the “Semiformally Designed and Tested” level.

4.4.2 Workload management under z/VM

z/VM has a capability of managing virtual machines. With *Virtual Machines Resource Manager (VMRM)* you can control the resources that one or more virtual machines are allowed to use. The VMRM was introduced in z/VM Version 4.3 and provides dynamic functions to:

- ▶ Manage guest performance

A *service virtual machine (SVM)* accepts customer-defined workload definitions, goal specifications, and associations between them. The SVM then adjusts virtual machine processor and I/O performance controls based on actual performance measurements to attempt to achieve the goals associated with each workload.

- ▶ Exploit I/O Priority Queueing

A virtual equivalent of the hardware I/O Priority Queueing facility allows virtual machines running guest operating systems such as z/OS that exploit I/O Priority Queueing to determine the priority of their I/O operations within bounds that can be defined on z/VM. z/VM will automatically set a priority for I/O operations initiated by virtual machines that do not exploit this function.

The VMRM has two implementations: one is called *VMRM Cooperative Memory Management (VMRM-CMM)* and the other is called *VMRM Collaborative Memory Management Assist (VMRM-CMMA)*. We discuss this in more detail in the next sections.

VMRM Cooperative Memory Management

VMRM-CMM assists in managing memory constraints in the system. Based on several variables obtained from the system and storage domain CP monitor data, VMRM detects when there is such constraint, and notifies a specific Linux virtual machine when this occurs. The virtual machine can then take the appropriate action to adjust its memory utilization in order to relieve this constraint on the system, such as issuing a CP DIAGNOSE X'10' instruction to release pages of storage.

In addition to the workload management functions for processor resources and DASD I/O provided by VMRM, the following is provided for Linux virtual machines:

- ▶ A NOTIFY statement with a MEMORY keyword in the Virtual Machine Resource Manager configuration file. Following the keyword is a user ID or a list of user IDs to be notified when virtual memory becomes constrained. For the format of this statement, see the NOTIFY Statement.
- ▶ System and storage domains are monitored for data to be used for calculating memory constraint, as well as how much memory to request the guest machine to release.
- ▶ When memory is constrained, VMRM issues a CP SMSG to notify the specified guests with the amount required to release in order to relieve the constraint. For the format of the SMSG buffer, see Usage Note 1.
- ▶ A message is logged in the VMRM log file, indicating which users were sent an SMSG, and the text of the SMSG buffer. Also, if MSGUSER is specified on the VMRM ADMIN

statement, the same message written to the log is written to the MSGUSER user ID's console as well.

VMRM Collaborative Memory Management Assist

The VMRM-CMMA is a machine feature that allows z/Architecture virtual machines with the appropriate support to exchange memory usage and status information with z/VM. This sharing of information provides several benefits:

- ▶ The guest memory footprint is reduced, allowing for greater memory overcommitment by z/VM.
- ▶ z/VM can make more efficient decisions when selecting page frames to reclaim.
- ▶ z/VM page-write overhead is eliminated for pages that the guest has designated as unused or volatile.
- ▶ The virtual machine can make better decisions when assigning pages.
- ▶ Virtual machine page-clearing overhead can be eliminated for pages that z/VM has indicated contain zeros.

The Collaborative Memory Management Assist provides a means for communicating state information about a 4 KB block of storage between a program running in a z/Architecture virtual machine and the z/VM control program. This sharing of state information allows the program and z/VM to make more efficient memory management decisions. The Collaborative Memory Management Assist includes the following features:

- ▶ A unique block-usage state and block-content state that are associated with each 4 KB block of main storage (that is, memory) in the virtual configuration.
- ▶ The privileged EXTRACT AND SET STORAGE ATTRIBUTES instruction which can be used to extract and optionally set the block-usage and block-content states of a 4 KB block.
- ▶ A new reason for recognizing an addressing-exception program interruption.
- ▶ The new block-volatility-exception program-interruption condition.

The principal advantage using VMRM is basically workload optimization and performance optimization. Figure 4-6 on page 68 shows studies in the laboratory³ comparing internal throughput rate and number of servers in some cases of implementation of VMRM and default configuration of virtual machines on z/VM 5.3.

³ See <http://www.vm.ibm.com/perf/reports/zvm/html/530cmm.htm>

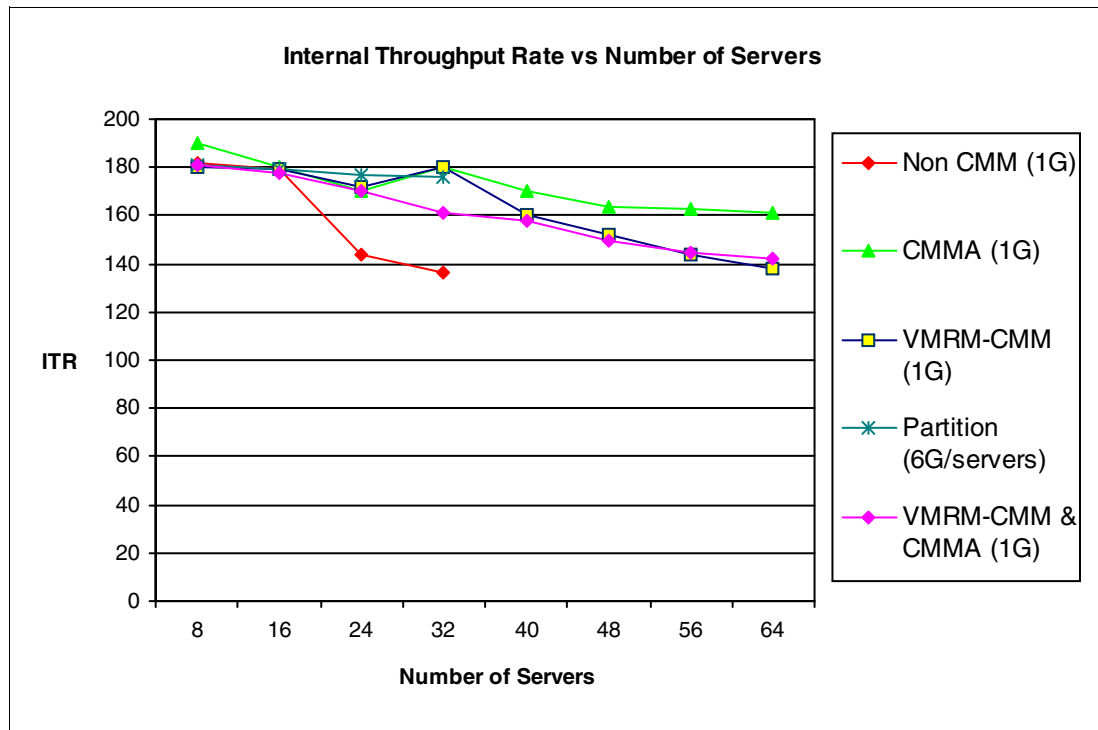


Figure 4-6 Laboratory studies using z/VM 5.3 and VMRM implementations

4.5 Network and communications

Many books could be filled with describing communications technology in System z, but we now focus only on a few aspects that you need to be aware of when considering migration of a workload to System z:

- ▶ Support of legacy protocols
- ▶ Server load distribution
- ▶ System z network connectivity

4.5.1 Support of legacy protocols

It is possible that the source application environment will employ some non-IP networking protocols. One example is the electronics payments arena and the existence of devices that require X.25 or SNA support. What about the network infrastructure? Will the target application environment need to support these protocols? Have the appropriate technologies been enabled to handle these protocols over the IP network, for example XOT (X.25 over TCP/IP) or SNA over IP?

Two publications we recommend if you have requirements to support legacy communication protocols:

- ▶ *IBM Communication Controller for Linux on System z V1.2.1 Implementation Guide*, SG24-7223
- ▶ *Enterprise Extender Implementation Guide*, SG24-7359

4.5.2 Server load distribution

A number of solutions are available to address server load distribution. During the migration effort there may be a need for the applications to run in parallel on both the source environment as well as the target System z environment. This would require load balancing at the router level to ensure that both the source and target migration environments received a share of the incoming workload. In the z/OS Parallel Sysplex environment, *Sysplex Distributor* is used to distribute incoming connections between LPARs as well as handle failovers.

Sysplex Distributor is a function in z/OS Communications Server that allows an IP workload to be distributed to multiple server instances within the sysplex. The sysplex consists of multiple z/OS images or LPARs and is viewed as a *dynamic virtual IP address (DVIPA)*. Sysplex Distributor is the internal decision-maker for application workload balancing (where the workload is TCP/IP connections). The default distribution method for Sysplex Distributor provides capacity recommendations in the form of weights for each system. WLM assigns a relative weight to each system in the sysplex, with the highest weight going to the system with the most available processor capacity. For further information, see *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 3*, SG24-7534.

4.5.3 System z network connectivity

IBM System z servers provide a wide range of interface options for connecting your system to an IP network or to another IP host. However, the following interfaces deliver the best throughput and performance, as well as offer the most flexibility and highest levels of availability:

- OSA-Express

OSA-Express utilizes a direct memory access (DMA) protocol to transfer the data to and from the TCP/IP stack. It also provides the offloading of IP processing from the host. The OSA-Express Ethernet features support IEEE standards 802.1p/q (priority tagging and VLAN identifier tagging). OSA-Express also provides primary (PRIRouter) and secondary (SECRouter) router support. This function enables a single TCP/IP stack, on a per-protocol (IPv4 and IPv6) basis, to register and act as a router stack based on a given OSA-Express port. Secondary routers can also be configured to provide for conditions in which the primary router becomes unavailable and the secondary router takes over for the primary router.

- HiperSockets

HiperSockets provides high-speed LPAR-to-LPAR communications within the same sever (through memory). It also provides secure data flows between LPARs and high availability, if there is no network attachment dependency or exposure to adapter failures.

HiperSockets connection supports *VLAN tagging*. This allows you to split the internal LAN represented by a single HiperSockets CHPID into multiple virtual LANs, providing isolation for security or administrative purposes. Only stacks attached to the same HiperSockets VLAN can communicate with each other. Stacks attached to a different HiperSockets VLAN on the same CHPID cannot use the HiperSockets path to communicate with the stacks on a different VLAN.

When the TCP/IP stack is configured with HiperSockets Accelerator, it allows IP packets received from HiperSockets to be forwarded to an OSA-Express port (or vice versa) without the need for those IP packets to be processed by the TCP/IP stack.

- Dynamic Cross-System Coupling Facility (dynamic XCF)

Cross-System Coupling Facility (XCF) allows communication between multiple CSs (Communications Servers) for z/OS IP stacks in a Parallel Sysplex. You have a choice of defining the XCF connectivity to other TCP/IP stacks individually, or using the dynamic XCF definition facility. Dynamic XCF significantly reduces the number of definitions that you need to create whenever a new system joins the sysplex or when you need to start up a new TCP/IP stack. These changes become more numerous as the number of stacks and systems in the sysplex grows. This could lead to configuration errors. With dynamic XCF, you do not need to change the definitions of the existing stacks in order to accommodate the new stack.

Fundamental technologies for z/OS TCP/IP availability

TCP/IP availability is supported as follows:

- Virtual IP Addressing (VIPA)

VIPA provides physical interface independence for the TCP/IP stack (the part of a z/OS Communications Server software that provides TCP/IP protocol support) and applications so that interface failures will not impact application availability.

- Static VIPA

A *static* VIPA is an IP address that is associated with a particular TCP/IP stack. Using either ARP takeover or a dynamic routing protocol (such as OSPF), static VIPAs can enable mainframe application communications to continue unaffected by network interface failures. As long as a single network interface is operational on a host, communication with applications on the host will persist.

- Dynamic VIPA (DVIPA)

Dynamic VIPAs (DVIPAs) can be defined on multiple stacks and moved from one TCP/IP stack in the sysplex to another automatically. One stack is defined as the primary or owning stack, and the others are defined as backup stacks. Only the primary stack is made known to the IP network.

TCP/IP stacks in a sysplex exchange information about DVIPAs and their existence and current location, and the stacks are continuously aware of whether the partner stacks are still functioning.

If the owning stack leaves the XCF group (because of some sort of failure), then one of the backup stacks automatically takes its place and assumes ownership of the DVIPA. The network simply sees a change in the routing tables (or in the adapter that responds to ARP requests).

- Address Resolution Protocol takeover

Address Resolution Protocol (ARP) enables your system to transparently exploit redundant physical interfaces without implementing a dynamic routing protocol in your mainframe. ARP takeover is a function that allows traffic to be redirected from a failing OSA connection to another OSA connection. If an OSA port fails while there is a backup OSA port available on the same subnetwork, then TCP/IP informs the backup adapter as to which IP addresses (real and VIPA) to take over and network connections are maintained. After it is set up correctly, the fault tolerance provided by the ARP takeover function is automatic.

- Dynamic routing

Dynamic routing utilizes network-based routing protocols (such as OSPF) in the mainframe environment to exploit redundant network connectivity for higher availability (when used in conjunction with VIPA).

- Internal application workload balancing

- Sysplex Distributor (SD)

The application workload balancing decision maker provided with the Communications Server is the Sysplex Distributor (SD). The design of the SD provides an advisory mechanism that checks the availability of applications running on different z/OS servers in the same sysplex, and then selects the best-suited target server for a new connection request.

The Sysplex Distributor bases its selections on real-time information from sources such as Workload Manager (WLM) and QoS data from the Service Policy Agent. Sysplex Distributor also measures the responsiveness of target servers in accepting new TCP connection setup requests, favoring those servers that are more successfully accepting new requests.

Internal workload balancing within the sysplex ensures that a group or cluster of application server instances can maintain optimum performance by serving client requests simultaneously. High availability considerations suggest at least two application server instances should exist, both providing the same services to their clients. If one application instance fails, the other carries on providing service. Multiple application instances minimize the number of users affected by the failure of a single application server instance. Thus, load balancing and availability are closely linked.

- Portsharing

In order for a TCP server application to support a large number of client connections on a single system, it might be necessary to run more than one instance of the server application. Portsharing is a method to distribute workload for IP applications in a z/OS LPAR. TCP/IP allows multiple listeners to listen on the same combination of port and interface. Workload destined for this application can be distributed among the group of servers that listen on the same port.

- External application workload balancing

With external application workload distribution, decisions for load balancing are made by external devices. Such devices typically have very robust capabilities and are often part of a suite of networking components.

From a z/OS viewpoint, there are two types of external load balancers available today. One type bases decisions completely on parameters in the external mechanism, and the other type uses sysplex awareness matrixes for each application and each z/OS system as part of the decision process. Which technique is best depends on many factors, but the best method usually involves knowledge of the health and status of the application instances and the z/OS systems.

- z/OS Parallel Sysplex

z/OS Parallel Sysplex combines parallel processing with data sharing across multiple systems to enable you to harness the power of plural z/OS mainframe systems, yet make these systems behave like a single, logical computing facility. This combination gives the z/OS Parallel Sysplex unique availability and scalability capabilities.

4.5.4 z/VM virtual networking

In this section we describe—at a high level—the networking facilities that z/VM provides to its guest virtual machines. The following topics are discussed:

- z/VM TCP/IP stack
- z/VM support of VLAN (IEEE 802.1Q)
- z/VM Virtual Switch (VSWITCH)

Each of these options provides a highly secure communication path, under control of the CP, that is not detectable or in any way “sniffable” by other virtual machines (that is, no other virtual machine can eavesdrop on the data moving between virtual machines). Of course, these virtual network connections are only as secure as the guests connected to them.

The z/VM TCP/IP stack

The z/VM TCP/IP stack is an optional service machine executing the TCP/IP stack task under CMS, which enables virtual machines hosted by the z/VM instance to be connected to external networks with some degree of protection against denial-of-service (DoS) attacks. The z/VM stack can be seen, from the networking standpoint, as a dual-homed TCP/IP stack—one network adapter being a physical adapter connected to the physical network and the other being a virtual one connected to a virtualized guest network or point-to-point connections (for example, virtual channel-to-channel). Figure 4-7 provides a simplified view of the topology, where the z/VM TCP/IP stack acts as a router between the external physical network and the virtualized networks.

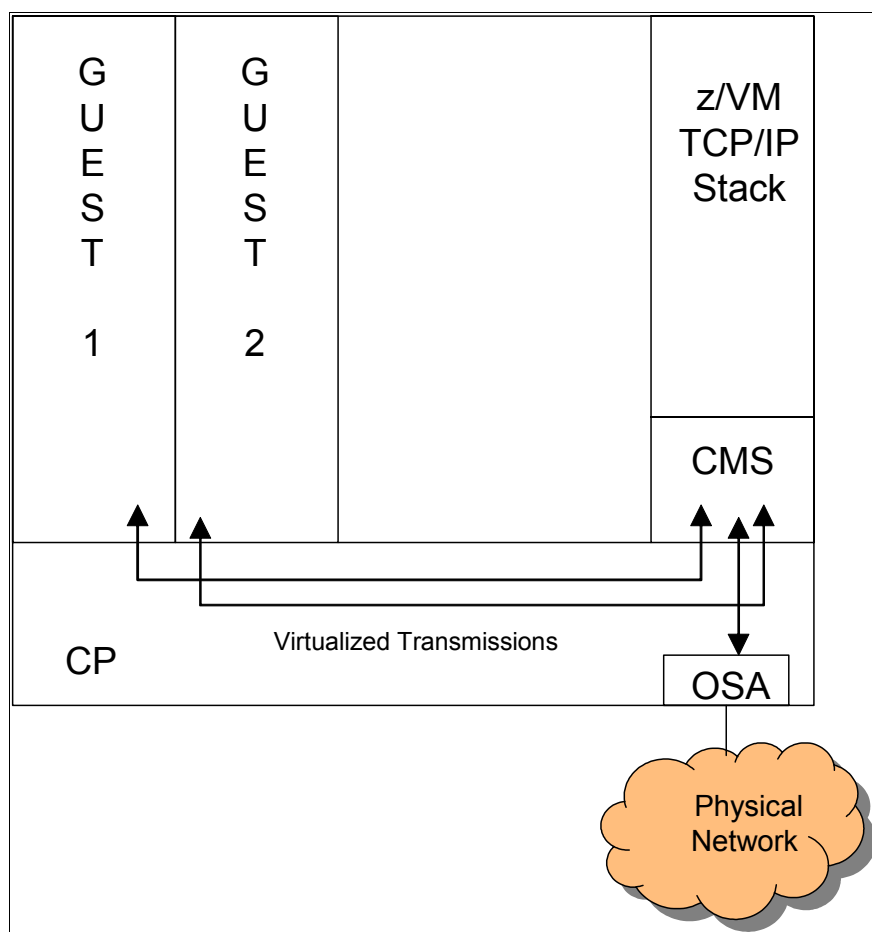


Figure 4-7 z/VM TCP/IP stack topology (simplified)

VLAN support

VLAN is a networking technique where the low-level Ethernet data frames conveyed by a physical network can be *tagged* (that is, additional information is added to each frame) as belonging to a specific Virtual LAN (VLAN). With this technique, multiple virtual LAN subnets can coexist on the same physical network. VLAN-aware adapters at the TCP/IP hosts connected to the physical network only respond to the traffic tagged for the VLAN to which they are connected. The VLAN specifications can be found in the IEEE 802.1Q standard.

Figure 4-8 provides a very high-level view of a network topology when the VLAN technique is used. Using VLANs requires having VLAN-aware routers, switches, and adapters connected to the network. In Figure 4-8 the switch accepts all regular or VLAN-tagged frames through the *hybrid* trunk port, then it routes or switches the tagged frames to the proper VLAN-aware devices on the basis of the VLAN numbers to which the frames and the devices belong.

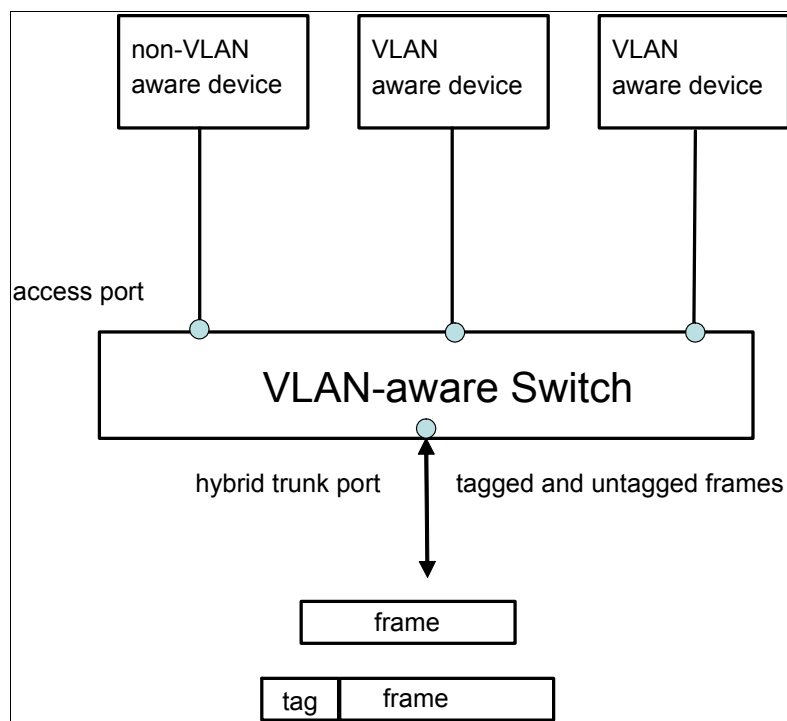


Figure 4-8 VLAN topology

VLANs facilitate easy administration of logical groups of machines that can communicate as though they were on the same LAN. However, beyond this implicit simplification of physical network implementation and administration, they also provide a degree of low-level security by restricting direct contact with a server to only the set of stations that comprise the VLAN.

On System z, where multiple stacks, in logical partitions or guest VMs, may exist potentially sharing one or more OSA-Express features, VLAN support is designed to provide a greater degree of isolation.

z/VM supports the IEEE 802.1Q standard for z/VM Virtual Switch, z/VM QDIO Guest LAN, and z/VM HiperSockets Guest LAN, allowing z/VM guests to create and participate in virtual LAN configurations.

z/VM Virtual Switch

The z/VM Virtual Switch is built on guest LAN technology and consists of a network of virtual adapters that can be used to interconnect guest systems. The virtual switch can also be associated with one or more OSA ports. This capability allows access to external LAN segments without requiring an intermediate router between the external LAN and the internal z/VM guest LAN.

The virtual switch can operate at Layer 2 (data link layer) or Layer 3 (network layer) of the OSI model and it bridges real hardware and virtualized LANs, using virtual QDIO adapters.

External LAN connectivity is achieved through OSA Ethernet features configured in QDIO mode. Like the OSA Ethernet features, the virtual switch supports the transport of Layer 2 (Ethernet frames) and Layer 3 (IP packets) traffic.

By default, the Virtual Switch operates in IP mode (Layer 3) and data is transported in IP packets. Each guest system is identified by one or more IP addresses for the delivery of IP packets. All outbound traffic destined for the physical portion of the LAN segment is encapsulated in Ethernet frames with the MAC address of the OSA port, as the source MAC address. With inbound traffic, the OSA port strips the Ethernet frame and forwards the IP packets to the virtual switch for delivery to the guest system based on the destination IP address in each IP packet.

When operating in Ethernet mode (Layer 2), the Virtual Switch uses a unique MAC address for forwarding frames to each connecting guest system. Data is transported and delivered in Ethernet frames. This provides the ability to transport both TCP/IP and non-TCP/IP based application data through the virtual switch. The address-resolution process allows each guest system's MAC address to become known to hosts residing on the physical side of the LAN segment through an attached OSA port. All inbound or outbound frames passing through the OSA port have the guest system's corresponding MAC address as the destination or source address.

The switching logic resides in the z/VM Control Program (CP), which owns the OSA port connection and performs all data transfers between guest systems connected to the virtual switch and the OSA port; see Figure 4-9.

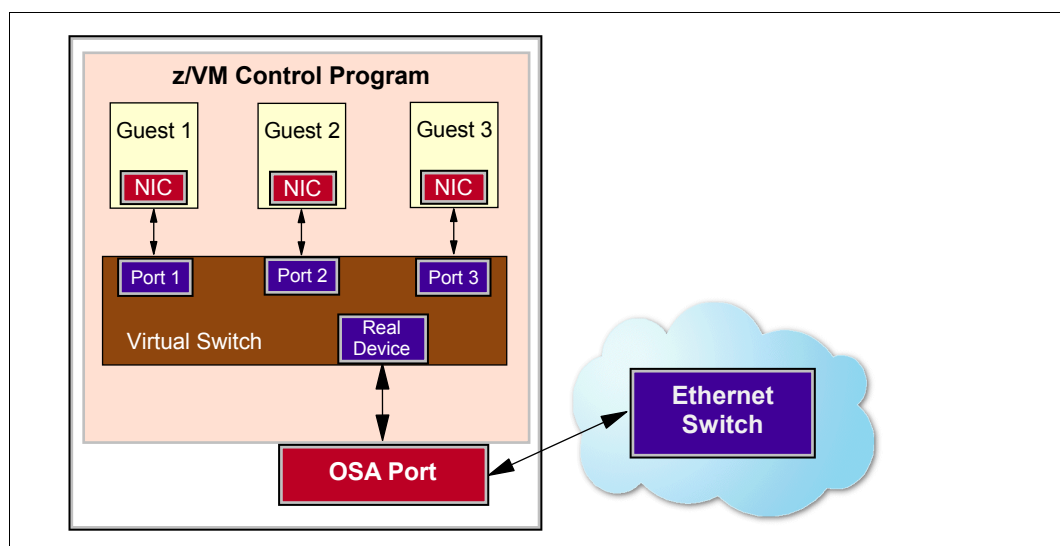


Figure 4-9 Data transfers between guest systems

4.5.5 Important decisions with respect to communications

With respect to communications on System z, the following decisions have to be made when adding a new workload to the environment:

- ▶ Does the workload require other communication protocols than TCP/IP and does the workload rely on other specific protocols?
Specific protocol support will need to be checked with the current System z specifications.
- ▶ Will the new workload on System z be able to benefit from HiperSockets?
This is the case if the workload is implemented on multiple LPARs on System z.

- If the workload is implemented in a Parallel Sysplex environment, is Sysplex Distributor and DVIPA a viable option to implement workload balancing across the LPARs?
- Is z/VM and running z/VM guests part of the solution?
If this is the case, the network has to be set up in a specific way to cope with multiple virtual servers.

4.6 Availability and scalability

In this section we discuss the key technologies available on System z to achieve high availability. In the broad sense of high availability, there must also be a solution for disaster recovery in place. Some technologies are specifically designed to achieve a solution for disaster recovery, other technologies are designed to meet SLAs during operation without a disaster occurring, and yet other technologies actually provide a solution for both.

4.6.1 Availability

We already talked about principles of high availability in other chapters in this book, but we briefly review some availability terms here.

High availability

High availability refers to maximum system uptime, and the level of high availability required for an environment is normally determined from a documented SLA. An environment that is designed to be highly available withstands failures that are caused by planned or unplanned outages.

Continuous operation

Continuous operation is the continuous, nondisruptive level of operation with the allowance of planned outages for changes to hardware and software that are transparent to users. Continuous operation environments are designed to avoid unplanned outages.

Continuous availability

Continuous availability is a continuous, nondisruptive level of service that is provided to users, and provides the highest level of availability that can possibly be achieved. In environments that are designed to provide continuous availability, planned or unplanned outages with hardware or software products cannot exist.

In most environments, provisions need to be made for planned outages to accommodate maintenance or upgrades to hardware or software. Therefore, continuous availability (in the purest sense) normally is not realistic, desirable, or even achievable. With the proper configuration, however, an environment can be achieved with near-continuous operation with high availability.

4.6.2 Disaster recovery

A disaster recovery situation occurs when there is an unplanned outage. Although the System z infrastructure that has been configured may consistently meet the high availability SLA, there is always the possibility, even in a continuous operations (not continuous availability) environment, that not enough redundancy has been built-in to prevent an outage, due to an unplanned event.

Any component in the end-to-end infrastructure that could cause the service to be interrupted due to an unplanned event, and for which there is no redundant component available to take over, is called a *Single Point of Failure (SPOF)*.

There are various methods for dealing with a disaster recovery situation. The method used in your environment will depend on the configuration and architecture that has been employed. While each operational model outlined in Chapter 5, “Operational models for high availability” on page 129 presents different challenges in maintaining high availability and recovering from a disaster, basic recovery scenarios are discussed here.

- ▶ One active and a second cold D/R site on a contract basis

In a configuration with one site, a contract can be purchased from a disaster recovery services organization (such as IBM Business Continuity and Recovery Services) at a “cold” site to build and recover your environment. In this scenario, you would typically plan disaster recovery exercises at the contracted cold site on a regular basis to ensure recovery procedures were still working properly to provide full recovery in a timely manner and were kept up to date to support the changing workloads.

- ▶ Two sites, one active and one passive for D/R

In an environment with two sites, using an active / passive configuration, there is effectively a “warm” backup site that would become operational in case of an unplanned outage in a primary site. Depending on the technology used, this can be a more or less automated procedure, but not necessarily. Also, in this scenario there will be some time lost until the second site is fully operational.

- ▶ Two sites, both active and each other’s D/R site

In a two-site scenario with either an active/active or dual active configuration, there is an active second location that can seamlessly pick up the load from the first site in a disaster situation.

Important: High availability comes with a cost. The question becomes: “How many single points of failure are you willing to eliminate when configuring your System z environment?” The higher the availability for which an environment is configured, the higher the price tag.

4.6.3 What about planned outages

Sometimes outages are necessary to maintain and upgrade hardware or software. These outages can be planned to minimize impact and continue to provide high availability and continuous operation. Planned outages can happen at different levels (LPAR, application server, processor, IO card, and so on) and you should determine at least the following for each planned outage:

- ▶ Does a planned outage occur in specific intervals?
- ▶ How long does the planned outage typically last?
- ▶ Is there a redundant component configured to take over?
- ▶ Is the redundant component live immediately, is it already active, or is time needed to bring it up?
- ▶ Do you need procedures, tools, programs or utilities to switch over to the redundant component?
- ▶ Do you need procedures, tools, programs or utilities to switch off the component that will be taken down?

In any case, recovery methodology employed for a particular configuration should be the same whether the outage is planned or unplanned.

In the next sections we discuss the primary technologies available on System z for high availability.

4.6.4 Parallel Sysplex

A Parallel Sysplex environment most probably offers the highest availability in terms of configuration. Redundancy of hardware and duplication of LPARs exist across System z servers in the same site or in multiple sites. Multiple sites would offer the most protection in terms of eliminating single points of failure (SPOFs). In any case, the utilization of each System z processor in a Parallel Sysplex environment should be maintained at 50% (or less) to allow for failover of workloads. Two Coupling Facilities (CFs) should be configured with structures to link duplicate or redundant LPARs and allow for data sharing to take place between these LPARs.

There are two flavors of CFs: internal and external. An *internal CF* is configured as another LPAR on the same System z server as other LPARs housing the applications. The internal CF utilizes one or more

specialty engines on the System z server and has memory assigned to it. An *external CF* is configured on a separate System z server that does not support other applications. In any case, utilization of each CF should be maintained at 50% or less in order to handle a possible failover situation. Typically the CFs (internal or external) have special channels that link the CFs to the LPARs in the Parallel Sysplex.

To improve recoverability and eliminate a CF single point of failure in case of an outage or failover situation, duplexing of structures should take place across the two CFs. This allows two copies of each structure to exist (one on each CF) and the contents of the structures to be synchronized in each copy. In the event of a CF failure or outage, the copies of the structures on the other CF are used to satisfy requests and support the applications.

Geographically Dispersed Parallel Sysplex (GDPS) is a multisite high availability solution that manages remote copy configurations and storage subsystems, automates Parallel Sysplex operational tasks, and performs failure recovery from a single point of control. In addition, it automates recovery procedures for planned and unplanned outages and helps provide near-continuous operation.

GDPS can also support a single site but that is not discussed here because it is more common for GDPS to operate in a multisite environment.

Parallel Sysplex architecture

Parallel Sysplex consists of a combination of hardware and software components. These provide a very advanced infrastructure that any database or application server providers (IBM and ISV) can use in a standardized manner to enable their applications to be shared across multiple systems with full data integrity and very high performance.

Table 4-2 on page 84 illustrates the Parallel Sysplex components in a two-way configuration sample. It is a logical representation of Parallel Sysplex. There are many variations of how the logical components can be mapped to physical hardware. In fact, z/OS images and CFs can run in LPARs as well as on physical CPCs.

For more information about Parallel Sysplex, refer to the Parallel Sysplex website at:

<http://www-03.ibm.com/systems/z/advantages/psa/index.html>

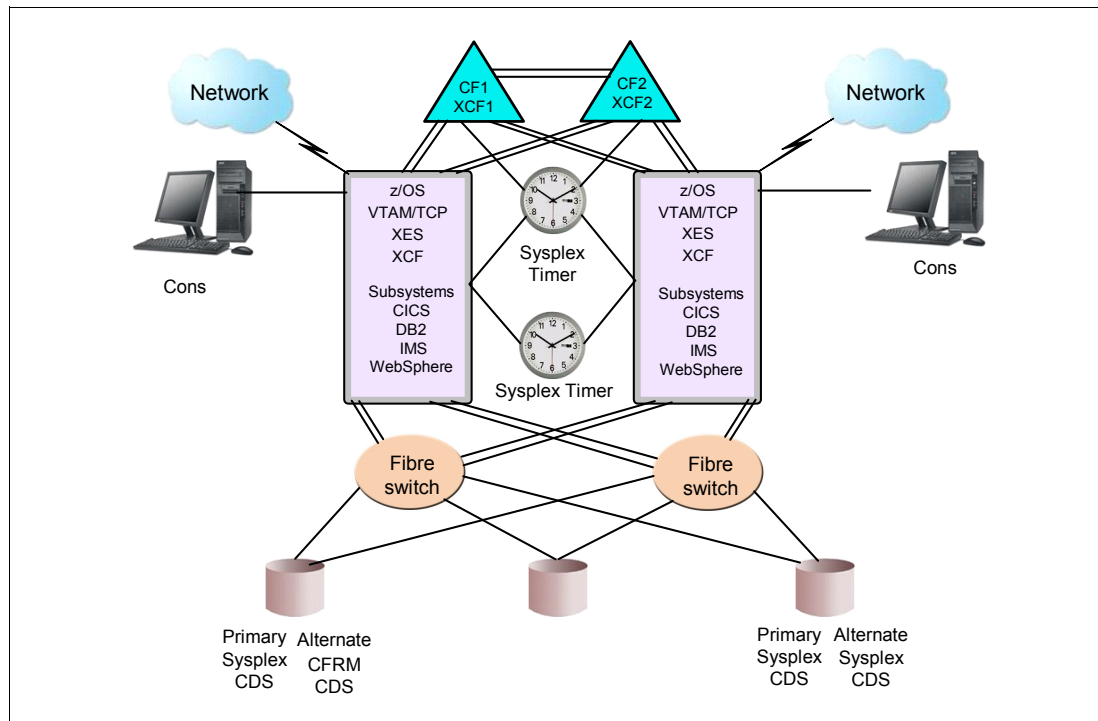


Figure 4-10 Parallel Sysplex architecture

4.6.5 High availability solutions for data

There are two main concepts for achieving high availability for data:

- Data sharing

Data sharing is a technique whereby multiple images of a system, either on the same server, in the same site or in multiple sites, have direct access to the same physical datastore. If a system image, or even an entire server or site, would fail, then another system image, server or site would still have access to the live datastore and be able to continue processing the workload. No procedure is needed to bring up another database or copy data over at the time of a failure. The only thing to worry about is to make sure pending commits on data updates are taken care of and that locks on data that is pending updates are eventually released.

System z's highest quality solution for data high availability is based on this technique and we discuss this further in "DB2 data sharing" on page 79.

- Data replication

In this category of solutions a redundant copy of the datastore is kept on another server or even in another site. To be able to keep that copy up to date, each data update on the primary server or site would have to be copied over to the datastore in the secondary (failover) server or site. These data replications can be synchronous or asynchronous. Synchronous provides the highest reliability, but may cause a performance delay in transactions.

The solutions available for data high availability on Linux on System z are using this technique and we discuss this further in "Data replication technologies" on page 79.

DB2 data sharing

Clients have many options when it comes to databases and file systems on System z. The primary Relational Database Management System (RDMS) on the System z platform is DB2 on z/OS. DB2 achieves the highest levels of availability, maintainability, resilience, and scalability through an optimized combination of hardware, operating system, APIs and tools. DB2 *data sharing* is an excellent example of leveraging these components in a Parallel Sysplex implementation.

DB2 data sharing takes advantage of the System z hardware and Parallel Sysplex technology to allow multiple different DB2 subsystems to read from and write to the same DB2 data concurrently. These different subsystems constitute a DB2 *data sharing group* and can be across multiple System z servers within the Parallel Sysplex cluster. One of the key components in DB2 data sharing is the *Coupling Facility (CF)*, which provides high-speed caching and locking functions. DB2 exploits the caching function by establishing *Group Buffer Pools* in the CF. These pools provide shared caches from which any member DB2 subsystem can quickly access data without requiring disk I/O. DB2 utilizes resource locking structures in the CF to coordinate simultaneous access to shared data. To provide redundancy, multiple Coupling Facilities can be configured within the Parallel Sysplex. The cache and lock structures stored on these CFs can be duplexed so that no cache or lock data will be lost in the event of a hardware failure.

DB2 exploits Parallel Sysplex technology to keep the application and subsystem state information in the Coupling Facility. This enables a “rolling window” update approach that allows members of the data sharing group to work seamlessly if one node fails. This provides an advantage that allows maintenance to be performed on a DB2 data sharing group member while the other DB2 group members remain online and active. All incoming work processes are routed to the remaining DB2 data sharing group members.

Data replication technologies

While DB2 data sharing is the recommended and most commonly used solution to achieve high availability with DB2 databases on z/OS, in Linux on System z high availability with databases is typically achieved with solutions based on replication. We discuss two solutions:

- ▶ Oracle Real Application Cluster (RAC)
- ▶ InfoSphere™ Change Data Capture (CDC)

Oracle RAC

Oracle RAC requires the use of *Automatic Storage Management (ASM)*.

ASM is a volume manager and a file system for Oracle database files that supports single-instance Oracle database and Oracle Real Application Cluster (RAC) configurations. ASM is Oracle's recommended storage management solution. It provides an alternative to conventional volume managers, file systems, and raw devices.

The main components of ASM are disk groups, each of which is comprised of several physical disks that are controlled as a single unit. The physical disks are known as ASM disks, while the files that reside on the disks are known as ASM files. The locations and names for the files are controlled by ASM, but user-friendly aliases and directory structures can be defined by the user for ease of reference.

The level of redundancy and the granularity of the striping can be controlled using templates. Default templates are provided for each file type stored by ASM, but additional templates can be defined as needed.

For a better understanding of performance using ASM on Linux on System z, see *Experiences with Oracle Solutions on Linux for IBM System z*, SG24-7634.

For more on Oracle ASM, visit:

<http://www.oracle.com/technology/products/manageability/database/pdf/asmov.pdf>

4.6.6 Storage resilience considerations

When analyzing availability at the LPAR, server, Parallel Sysplex or site level, DASD storage also needs to be examined from a resilience or recoverability perspective. As an introduction, some available remote copy technologies need to be discussed:

- ▶ Flashcopy

Copies full volumes of data in a storage subsystem. It takes only a few seconds to perform flashcopies of several volumes.

- ▶ Global Copy

Copies data over virtually unlimited distances asynchronously. With the copy the source volume periodically sends an incremental copy of updated tracks to the target volume instead of sending a constant stream of updates.

- ▶ Metro Mirror (PPRC)

Mirrors disk volumes within one site or between 2 sites that can be up to 300 km apart. This is a synchronous copy solution using *Peer to Peer Remote Copy (PPRC)* technology.

- ▶ z/OS Global Mirror (XRC)

Copies disk volumes from the local site to the remote location without the 300 km distance limitation. It is an asynchronous copy solution using extended remote Copy (XRC) technology that utilizes the DFSMS *System Data Mover (SDM)*. Note that SDM has the ability to run on the zIIP specialty engine.

- ▶ Global Mirror (PPRC)

Provides disk-to-disk copy between two sites over virtually unlimited distances. It uses two other disk copy functions (Flashcopy and Global Copy) with PPRC asynchronous technology.

From a high availability and performance perspective, synchronous as opposed to asynchronous remote copy functions are more desirable since the copy function is immediate and therefore more timely. However, locations of the primary and secondary disk subsystems have to meet distance criteria.

With a single site, there should be no copy issues if redundant subsystems are not warranted or justified. All LPARs can be connected to the same DASD subsystems. Most DASD subsystems have high reliability because of redundancy of hardware components, but failures have been known to happen on occasion.

If redundant disk subsystems are warranted in a single-site configuration, then the basic hyperswap function can be implemented to simplify the management of remote copy configurations. Hyperswap is currently supported by the IBM TotalStorage® ESS800, DS6000™ and DS8000® families of disk subsystems utilizing Metro Mirror (PPRC) synchronous remote copy technology. In planned or unplanned outage situations, hyperswap allows swapping of primary to secondary disk subsystems in a matter of seconds. A planned outage could be scheduled to perform disk maintenance or practice a DR exercise. The switching to secondary disk subsystems could then be done without quiescing the applications. In an unplanned outage scenario, hyperswap could mask primary disk subsystem failures by transparently switching to the secondary disk subsystem.

With a dual site that is not configured as a Parallel Sysplex environment, Metro Mirror (PPRC), z/OS Global Mirror (XRC) or Global Mirror (PPRC) technology could be used to

provide disk copy services between the two sites, depending on the distance between the two sites and response time and performance requirements for the copy functions. In addition, for an environment where DB2 data replication is taking place, InfoSphere Data Replicator could be used to perform remote copy services. Infosphere uses MQ technology to replicate data between DB2 databases in either a unidirectional (for active/passive) or bidirectional (active/active) configuration. InfoSphere could also be used for data replication in a Linux on System z environment.

For Parallel Sysplex environments, Geographically Dispersed Parallel Sysplex (GDPS) can be configured and GDPS options can be used between multiple sites to provide disk replication. These options are all based on the copy technologies discussed above, namely, Metro Mirror (synchronous PPRC), z/OS Global Mirror (XRC) or Global Mirror (asynchronous PPRC). In particular, some of the GDPS remote copy options available are:

- ▶ GDPS/PPRC with Metro Mirror (synchronous) and Hyperswap could be used technically between two sites that are less than 300 km apart. Realistically, however, this technology should only be used if distances are less than 100 km because of I/O response time and performance issues due to the synchronous remote copy taking place.
- ▶ GDPS/XRC is a disaster recovery solution between two sites separated by virtually unlimited distance. This solution is based on the z/OS Global Mirror (XRC) technology and uses asynchronous remote copy.
- ▶ GDPS/Global Mirror is also a disaster recovery option between two sites without distance limitations. This solution is based on the Global Mirror asynchronous PPRC remote copy technology.

Specific considerations for Linux on System z

When we consider high availability of Linux running on System z in the context of storage resiliency, you need to consider the same issues as discussed earlier with regards to z/OS and DASD volumes. If Linux on System z is considered, solutions involving open storage and Storage Area Network (SAN) should also be explored. Linux on System z can connect to a network and access the SAN volumes defined in an open storage subsystem.

For basic operation, in a high availability environment, the recommendations for basic connectivity must be followed: the recommendations are to use two or more channels for access to each volume, the use of SAN switches with alternative channels of access (if using open volumes), and use and configuration of multipathd on Linux on System z (if you use open storage).

Open storage

When using open storage devices on Linux on System z, you need to use multipathd to deliver a highly available environment.

Example 4-1 on page 82 shows a sample of output of the **multipath -ll** command on Linux on System z.

Example 4-1 Multipath output

```
[root@server01 etc]# multipath -ll
disk0-006 (36005076306ffc5290000000000001005) dm-148 IBM,2107900
[size=9.0G][features=1 queue_if_no_path][hwhandler=0][rw]
\_ round-robin 0 [prio=4][active]
  \_ 1:0:1:3   sdfs 130:224 [active][ready]
  \_ 0:0:1:3   sdf  8:80   [active][ready]
  \_ 2:0:1:3   sdmf 69:368 [active][ready]
  \_ 3:0:1:3   sdss 8:512  [active][ready]
disk1-126 (36005076306ffc529000000000000110e) dm-102 IBM,2107900
[size=9.0G][features=1 queue_if_no_path][hwhandler=0][rw]
\_ round-robin 0 [prio=4][active]
  \_ 0:0:1:63   sddv 71:208 [active][ready]
  \_ 1:0:1:63   scki 66:352 [active][ready]
  \_ 2:0:1:63   sdqv 132:496 [active][ready]
  \_ 3:0:1:63   sdxl 71:640 [active][ready]
```

In this sample, we have four (04) paths for each device. If one path fails, we have other paths that guarantee the availability of the device and the data access. With **multipathd** different policies can be used. The policy above is called *multibus*. Another policy would be *failover*, in which case we would have one path active and all others inactive, waiting for an event of unavailability on the active path to take control.

For a complex environment of highly available storage and Linux on System z, there are various solutions that guarantee this availability.

- Geographically Dispersed for Open Clusters (GDOC)

GDOC is an IBM Services implementation that addresses the automation, testing, and management requirements of a disaster recovery. It is designed to provide you with similar functionality for open systems that GDPS provides for the IBM System z mainframe. GDOC considers Open Storage and can be used if the client uses Open Storage on Linux on System z. GDOC is also available for other platforms such as AIX®, Hewlett-Packard UX, Linux, Microsoft Windows, Solaris, and Red Hat Linux and SUSE Linux.

- GDPS/PPRC and GDPS/Global Mirror

Geographically Dispersed Parallel Sysplex is also available for Linux on System z when Linux is using DASD devices. The implementation is similar to the z/OS implementation.

4.6.7 Scalability

Scalability is the ability to grow an environment to support increased workloads. Although there are many types of scalability, we will address horizontal, vertical, and depth scalability here.

Horizontal scalability involves adding new machines to a Parallel Sysplex to absorb anticipated growth in the same site or in different sites while *vertical scalability* involves adding resources (processor, memory, and so on) on an existing machine. *Depth scalability*, on the other hand, involves adding LPARs on an existing System z server using existing excess capacity and resources.

To achieve vertical scalability on an existing System z server, general CPs or specialty engines (such as zIIPS or IFLs) can be added to service workload growth or migration of work from another platform. They can be added via planned upgrades in a linear fashion up to the maximum number of CPs or mixture of CPs and specialty engines allowed on the System z server. See "Specialty processors" on page 124 for more information.

Increasing CPs

Capacity on Demand (CoD) is another form of vertical scalability available on a System z server. It allows extra hardware (CPs) that is currently configured but offline to the System z server to be brought online dynamically to absorb growth or absorb peak workloads. No outage or planned upgrade needs to occur to take advantage of the increase in capacity.

The functions of Capacity on Demand are designed to provide more flexibility and to provide an easier way to dynamically change capacity when business requirements dictate. Capacity on Demand is the key feature that enables you to adjust your processing and memory capacity to your specific needs without having to shut down or restart the server, and without needing to re-IPL the operating system.

Additional capacity resources can be dynamically activated, either partially or in total, by using granular activation controls directly from the management console of the System z server, without having to interact with IBM Support. CoD delivers *permanent* or *temporary* capacity upgrades.

► Online permanent upgrade

Up to the limits of the installed capacity on an existing server, a permanent upgrade can concurrently add processors, add memory, and change the subcapacity setting.

A permanent upgrade is initiated by the client using the *Customer Initiated Upgrade (CIU)* facility on the IBM Resource Link™. The CIU facility is the IBM online infrastructure that enables you to order permanent and temporary upgrades for a System z server. A permanent upgrade requires a contract to be signed. This contract covers the permanent upgrade buying capability.

► Temporary upgrades

Temporary upgrades can be done by *Capacity Backup (CBU)*, *Capacity for Planned Events (CPE)*, or *On/Off Capacity on Demand (On/Off CoD)*, all by using the CIU facility on the IBM Resource Link.

- Capacity Backup (CBU) is intended to replace capacity lost in the enterprise as a result of a disaster. CBU cannot be used for peak workload management. A CBU can last up to 90 days when a disaster situation occurs.
- Capacity for Planned Events (CPE) is used to replace temporary lost capacity in an enterprise for planned downtime events, for example for data center changes. CPE is not meant to be used for peak workload management, or for a disaster situation. The CPE provides for one 72-hour activation.
- On/Off Capacity on Demand (On/Off CoD) enables *concurrent* and *temporary* additional capacity on the server. On/Off CoD can be used for client peak workload requirements. It has a daily hardware charge. The software charges vary according to the license agreement for the individual products.

Figure 4-11 on page 84 shows an overview of CoD. Processor capacity consists of purchased and dormant capacity. Purchased capacity consists of active and inactive processors. CoD offerings can activate dormant and purchased processors, but inactive processors (in case of CPs) change the capacity level of purchased and active processors.

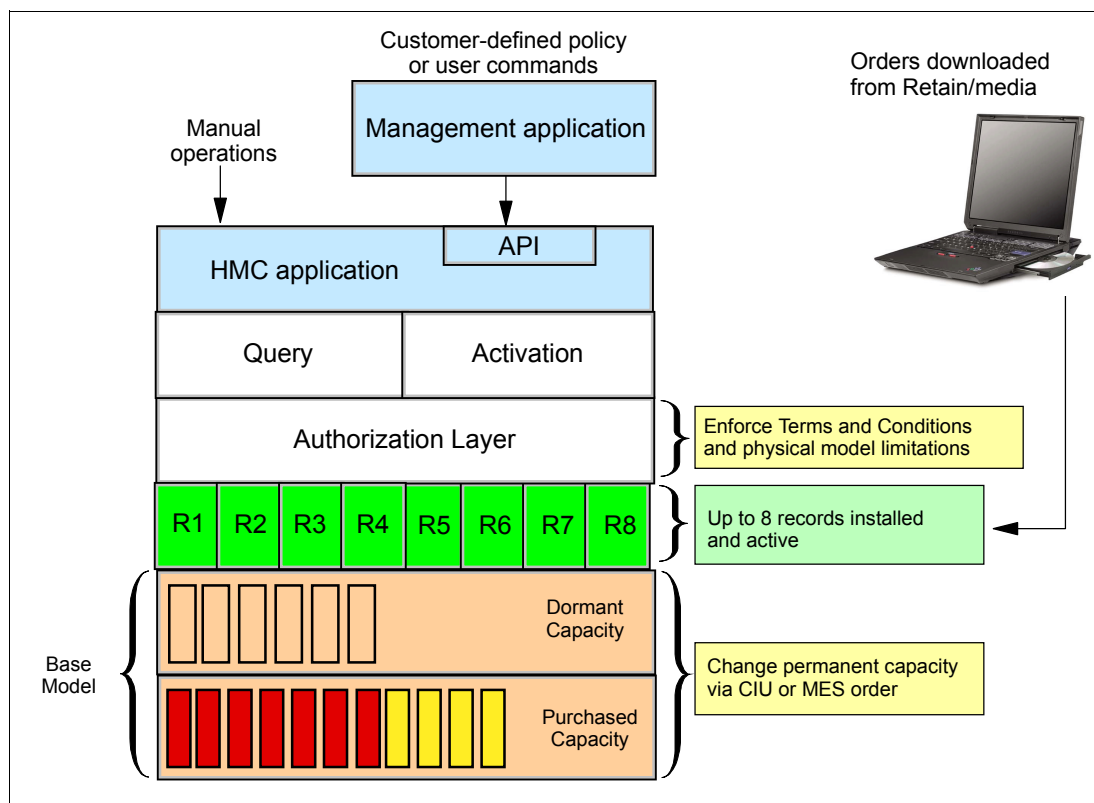


Figure 4-11 Capacity on Demand

The Capacity on Demand offerings are important features to meet client requirements for more flexibility, granularity, and better business control over the infrastructure, both operationally and financially.

Increasing memory

Memory (central storage) can also be added to an existing System z server to achieve vertical scalability. However, there are guidelines in terms of the granularity of adding storage which are dependent on the total size of physical storage that currently exists on the machine. Current guidelines for adding storage on the z9®, z10™ and z196 processors are reflected in Table 4-2, Table 4-3 on page 85, and Table 4-4 on page 85.

Table 4-2 z196 storage add granularity

increment	limitation on processor
32 GB	Up to 256 GB
64 GB	Above 256 GB up to 512 GB
96 GB	Above 512 GB up to 896 GB
112 GB	Above 896 GB up to 1008 GB (1 TB)
128 GB	Above 1008 GB up to 1520 GB
256 GB	Above 1520 GB up to 3056 GB (3 TB)

Table 4-3 z10 storage add granularity

increment	limitation on processor
256 MB	Up to 128 GB
512 MB	Above 128 GB up to 256 GB
1 GB	Above 256 GB up to 512 GB
2 GB	Above 512 GB up to 1 TB

Table 4-4 z9 storage add granularity

increment	limitation on processor
64 MB	Up to 32 GB
128 MB	Above 32 GB up to 64 GB
256 MB	Above 64 GB up to 128 GB
Same as z10	Above 128 GB

Vertical scalability can also occur within a System z server on an LPAR basis, utilizing existing resources that have been defined in reserve with the LPAR definition parameters (see "LPARs and LPAR management on System z" on page 53). Additional CPs as well as more memory can be varied online with console commands for any given LPAR with the proper LPAR definition parameters in place.

4.6.8 Important decisions with respect to availability and scalability

We already discussed high availability extensively in other chapters of this book. With respect to availability and scalability on System z the following decisions have to be made when adding a new workload to the environment:

- ▶ Which target architecture is needed to fulfil the SLA?
In "Target reference architectures" on page 16 we discuss the possible high availability and disaster recovery infrastructures. The choice depends on how much risk and the cost associated with that risk one is willing to take due to an outage.
- ▶ Which System z operational model is the best fit for the new workload?
Refer to Chapter 5, "Operational models for high availability" on page 129 for the possible options.

4.7 Security

Security is a broad topic and has many dimensions, and it is not possible to explain and describe all available security technologies on System z in this book. We will attempt to mention and describe the most important security aspects on System z from the perspective of migrating a workload from another platform to System z. These topics are:

- ▶ Certifications
- ▶ Encryption
- ▶ User identification and authentication
- ▶ Auditing and logging

- ▶ Digital certificate hosting
- ▶ Directory services

We discuss these topics for z/OS, z/VM, and Linux on System z in the following sections.

4.7.1 Security in z/OS

The IBM Redbooks publication *Security on the IBM Mainframe*, SG24-7803 was recently published with an extensive overview of security functionality in z/OS(). Therefore we do not repeat this information in this book. However, we did decide to include a section on address space security only.

Address space protection in z/OS

The z/OS operating system employs *address spaces* and *storage-protect keys* to further protect key programs within an LPAR from corrupting each other's private storage or data areas. It helps ensure that z/OS provides the maximum in data integrity and availability.

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.

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.

The virtual address space is divided into *private* and *common* areas according to the purpose of their use.

- ▶ 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 (zero). 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.

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 real storage is then protected from unauthorized users by a double protection mechanism, namely the protection key and the fetch bit.

A control field called a *protection key* is associated with each 4 KB frame of real storage. When a request is made to modify or read the contents of a real memory location, the key associated with the request is compared to the memory protect key and access is allowed if

the keys match, or if 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, the request is approved or an exception interrupt occurs.

4.7.2 Security in z/VM and Linux on System z

In the following sections, we discuss the security aspects of z/VM and introduce how z/VM virtualization can provide a secure isolation between guests on System z. We also discuss the Resource Access Control Facility (RACF) and Lightweight Directory Access Protocol (LDAP) on z/VM and how you can use them to allow an enterprise-wide point of control.

z/VM's role today is that of a hypervisor operating system. Virtual machine design began in the early 1960s, when IBM was exploring how to meet customer expectations using virtualization. The development of virtual machines was closely tied to the development of virtual storage, because they needed to operate together. The core of z/VM (that is, the hypervisor) is the *control program*. The control program creates and maintains virtual environments for virtual machines (guests). Figure 4-12 represents a z/VM system with multiple guests.

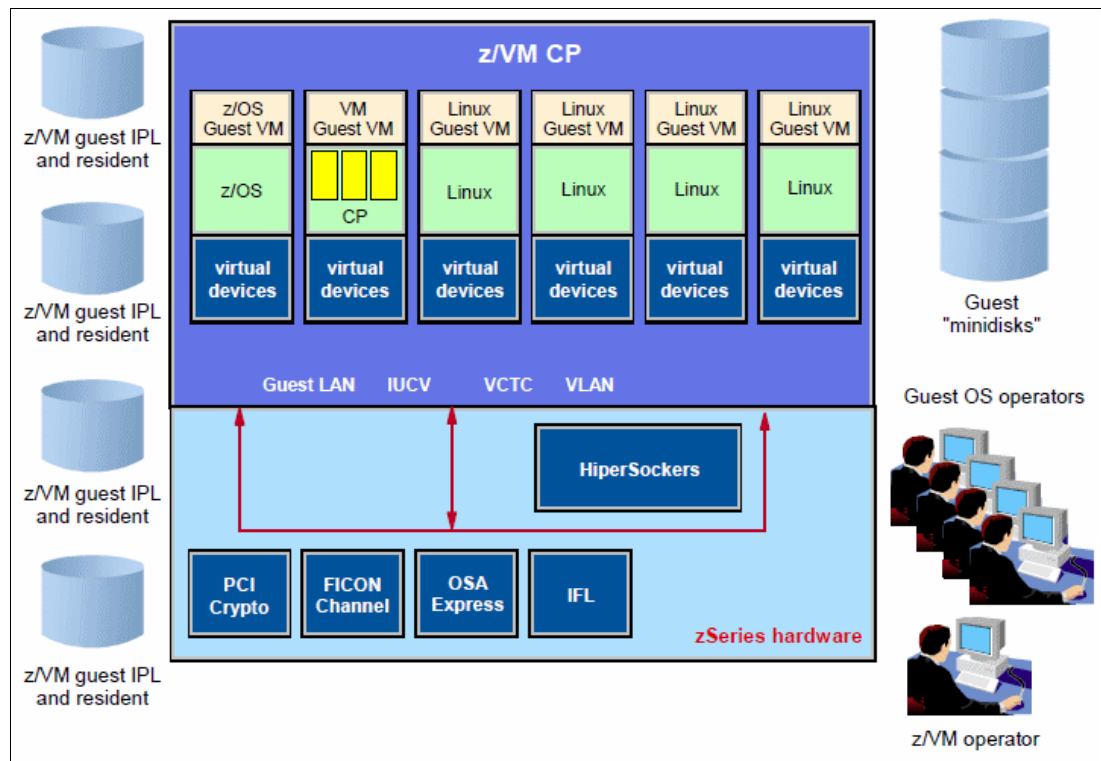


Figure 4-12 z/VM implementation with multiple guests

Referring to Figure 4-12, note the following:

- ▶ Only the control program is IPLed using the hardware IPL sequence. The guest IPL sequence is simulated by the control program.
- ▶ The control program operator console provides control program emulated *hardware consoles* for the guest virtual environment. Control program commands, on top of the guest IPL command, provide, for example, the equivalent of a *system reset* or *restart* hardware functions for the guest machines.

- ▶ The operating systems running in each guest have their usual operator consoles, which are physically connected through I/O channels to their respective operating systems.
- ▶ z/VM has its own scheme of disk storage partitioning, known as *minidisks*, that the control program uses to share one physical disk among several guests.
- ▶ z/VM can itself be running in a guest virtual machine, thus creating *second-level* guest z/VMs.
- ▶ The security requirements, as seen from the z/VM software perspective, ensure that the guest z/VMs have access only to the physical resources to which they are entitled and to guarantee inter-guest isolation. Each guest operating system is then responsible for its security environment as seen by its own users.

Note: These control program and guest software layers created in z/VM exploit the z/Architecture and use the physical processors by switching instruction flows and address spaces between the control program, guest operating systems, and user programs. They all exploit the same System z instruction set, with the exception of the control program. The control program uses a very specific instruction that is designed for virtual machine use called the *start Interpretive Execution (SIE)*.

z/VM certification

The *Common Criteria* is an internationally recognized International Standards Organization (ISO) standard that is used by governments and other organizations to assess the security and assurance of technology products. Under the Common Criteria, products are evaluated according to strict standards for various features, such as security functionality and the handling of security vulnerabilities.

The LSPP security labeling system

Central to Labeled Security Protection Profile (LSPP) security is its system of security labeling. Each object in a z/VM LSPP-compliant system has a *security label*, or *SECLABEL*, that designates its relative confidentiality and its membership in a security category. An object's security label defines what sort of data it can contain and, by implication, what sort of data it cannot contain.

Mandatory Access Control (MAC) is a security policy that governs which subjects can access which objects, and in what way, based upon certain rules. These rules are the **-property* and the *simple security property*. RACF commands are used to manage MAC for control program commands, DIAGNOSE codes, and system functions. MAC restricts a subject's access to an object.

CAPP

The *Controlled Access Protection Profile (CAPP)* specifies a set of security functional and assurance requirements, including access controls that are capable of enforcing access limitations on individual users and data objects. CAPP-conforming products also provide an audit capability that records the security-relevant events that occur within the system. CAPP was derived from the requirements of the C2 class of the U. S. Department of Defense Trusted Computer System Evaluation Criteria (TCSEC), dated December 1985. This protection profile provides security functions and assurances that are equivalent to those provided by the TCSEC and replaces the requirements used for C2 trusted product evaluations.

In z/VM, the CAPP requirements are met through the following specific mechanisms:

- ▶ *Discretionary Access Control (DAC)*

A method of restricting access to data objects based on the identity of users or groups to which the users belong. DAC protects system objects from unauthorized access by any user. Normally, permission to access an object is granted by the owner of the object. Occasionally, it can be granted by someone else, such as a privileged administrator.

► **Auditability of security-relevant events**

The recording of facts that describe a security-relevant event taking place in a computing system. In general, a security-relevant event is one that occurs in a computing system that, for better or for worse, affects the safety and integrity of the system's processes and data. The facts recorded that describe such an event include the time and date of the event, the name of the event, the name of the system objects affected by the event, the name of the user who caused the event to occur, and additional information about the event. In general, the security-relevant events in z/VM are:

- CP commands
- DIAGNOSE functions
- Communication among virtual machines

► **Object reuse**

A practice that prevents any newly assigned storage object from making available to its new owner any data that belonged to its former owner. This includes any encrypted data. Object reuse also requires the elimination of any residual user authorization access to a previously existing object. This ensures that if another, new object occurs in the system later under the same name, the subjects with access to the old object will not have access to the new one.

► **Identification and authentication**

A method of enforcing individual accountability by providing a way to authenticate a user's identity uniquely and unambiguously. Thus, any security-relevant action that users might take can be attributed to them.

z/VM V5.1 was evaluated for conformance with the Controlled Access Protection Profile (CAPP) and the Labeled Security Protection Profile (LSPP) of the Common Criteria, both at Evaluation Assurance Level (EAL) 3+.

z/VM V5.3 was evaluated with the RACF Security Server optional feature for conformance to the Controlled Access Protection Profile (CAPP) and LSPP of the Common Criteria standard for IT security, ISO/IEC 15408, at Evaluation Assurance Level 4, augmented by flaw remediation procedures (EAL4+). This satisfies the statement of direction made in the software announcement dated February 6, 2007.

All certification activities for z/VM V5.3 are complete. The certifying body issued its certification on July 28, 2008. z/VM V5.3 with the RACF Security Server optional feature has been certified to conform to the Controlled Access Protection Profile (CAPP) and Labeled Security Protection Profile (LSPP) of the Common Criteria standard for IT security, ISO/IEC 15408, at Evaluation Assurance Level 4+ (EAL4+).

For z/VM V6.1 with the RACF Security Server optional feature, including labeled security, for conformance to the Operating System Protection Profile (OSPP) of the Common Criteria standard for IT security, ISO/IEC 15408, at Evaluation Assurance Level 4 (EAL4+). Note: This statement of direction was made in a July 22, 2010 IBM announcement for z/VM V6.1. All statements regarding IBM's plans, directions, and intent are subject to change or withdrawal without notice.

For more details about certification, visit:

<http://www.vm.ibm.com/security/>

Encryption on z/VM and zLinux

The following sections discuss encryption in z/VM and Linux on System z.

Secure communication to z/VM using SSL

Depending on the security policies in an enterprise, and depending on the network environment, clients might want to secure (encrypt) the communication for their connections to the z/VM guest user IDs to avoid sending passwords in clear text over a network and to protect the content of the communication.

With z/VM you can set up the TCP/IP connections to the z/VM guests to be protected by *Secure Socket Layer (SSL)*.

As in other areas, the implementation of SSL is done in z/VM by using a virtual server to handle the work. The SSL server, which runs in the SSLSERV virtual machine, provides processing support for encrypted communication between remote clients and a z/VM TCP/IP server that listens on secure ports. The SSL server manages the database in which the server authentication certificates are stored. The TCP/IP stack server routes requests for secure ports to the SSL server. The SSL server, representing the requested application server, participates in the handshake with the client in which the cryptographic parameters are established for the session. The SSL server then handles all the encryption and decryption of data. Figure 4-13 illustrates the principal setup and the information flow.

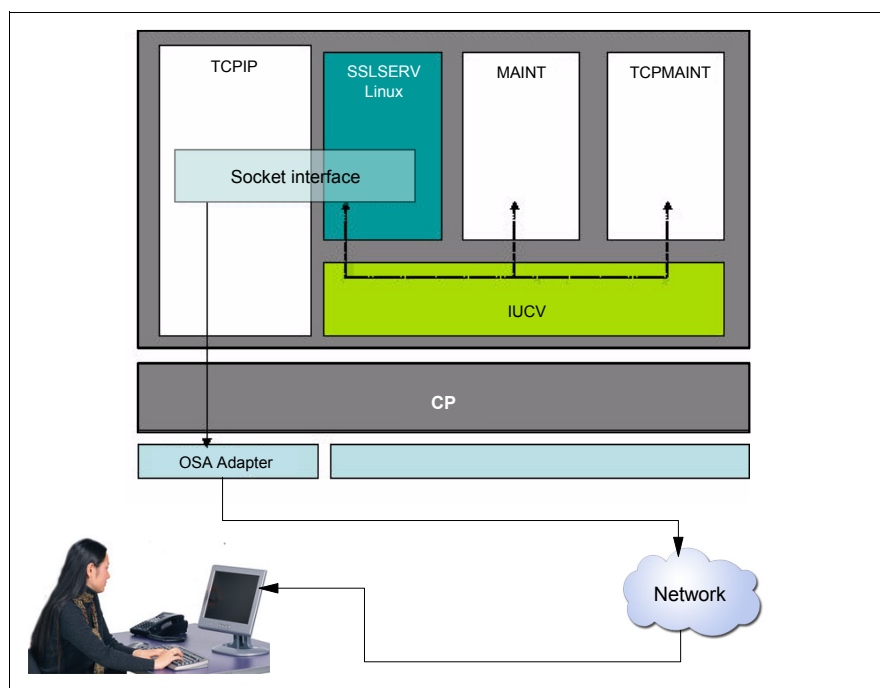


Figure 4-13 SSL implementation architecture in z/VM

The SSL Server is a virtual service machine with a special Linux server installed and configured for exclusive use of SSL. Only specific Linux distributions and kernel levels are supported. For a list of these, and also for detailed setup instructions, see the z/VM web page at:

<http://www.vm.ibm.com/related/tcpip/>

On this page you can find a link to SSL Server configuration where all related information is available:

<http://www.vm.ibm.com/related/tcpip/vmsslinfo.html>

Verify whether there are important service updates that might contain new information such as necessary PTFs or APARs. Check this information before you begin your implementation.

During the SSL handshake the client and the SSL server negotiate which encryption algorithms will be used to secure the connection. A cipher suite is selected by them that is common for both parties. Using the **ssladmin query status** command in the TCP/IP server, you can verify which cipher suites are allowed to be used by the SSL connection. Note that not all cipher suites provide a high degree of security. We recommend that you carefully consider which suites to allow. You might want to exempt individual cipher suites, such as NULL, NULL_SHA, or NULL_MD5, or you might want to instruct the SSL server to operate in Federal Information Processing (FIPS) mode.

The SSL support of z/VM is an easy method to protect the communication to the z/VM server, especially for administration tasks. This increases the total security and protection of the z/VM system, because sensitive information, such as passwords from administrators, is protected independent of the network.

Enhanced SSL server on z/VM V5R4

With the PTF for APAR PK65850 (for additional required service, see the VM540 subset of the TCPIP540 PSP bucket), z/VM V5.4 provides an SSL server, which runs in a CMS environment, replacing the Linux-based SSL server provided in previous z/VM releases. This change removes the requirement of using a Linux distribution to host the SSL server. The CMS-based SSL server might enable encryption services to be deployed more quickly and can help make installation, service, and release-to-release migration simpler. Other enhancements to the z/VM SSL server include:

- ▶ **Network-free SSL server administration**

The SSL server can be managed without requiring a network connection between the SSL server administrator and the SSL server.

- ▶ **New encryption and decryption engine**

The SSL server uses z/OS V1.10 System SSL technology for encryption, decryption, and certificate management.

- ▶ **New certificate-management services**

The System SSL **gskkyman** utility is now used to manage the SSL server certificate database. New services available for the SSL server include certificate renewal, certificate signing, and certificate exportation with or without the private key. The gskkyman application also manages certificates for the z/VM LDAP server.

Cryptographic software support in z/VM

Virtual machines enable the sharing of System z hardware among many operating systems. As a virtualization solution, the z/VM operating system does not provide a direct interface into any of the cryptographic hardware, nor does it require cryptographic services itself. z/VM provides a means of sharing the hardware cryptographic resources among the operating systems that are hosted (known as guests). Cryptographic resources can be shared through z/VM as follows:

- ▶ For all available cryptographic accelerators, z/VM provides unlimited access to all guests. This includes access to the CP Assist Cryptographic Function (CPACF) and the CEX3C⁴ when configured as a pure accelerator (CEX3A⁵).

⁴ Crypto Express3 co-processor

- The secure key Hardware Security Module (HSM), also sometimes referred to as a Tamper Resistant Security Module (TSRM), comprises a highly specialized piece of equipment that is designed to be the basis of your cryptographic security solution. These devices are the strongboxes that protect your symmetric keys and our asymmetric private keys. For HSM processors (CEX3C), z/VM can assign them to guests as well, but like logical partitioning, z/VM must assign the domains to each guest. A guest can have more than one domain. Also, multiple guests can be assigned the same domain, but only one guest can be active in a domain at any time.

For z/VM TCP/IP prior to z/VM Version 6.1, z/VM TCP/IP provides SSL support through a program interface called VMSOCK. Calls are redirected to a Linux on System z guest under z/VM, which contains special code provided by z/VM. This implementation was available for TN3270 or programs that were written to take advantage of the interface.

With z/VM Version 6.1, the usage of the SSL was extended to Transport Layer Security (TLS) and now provides full support for TN3270, SMTP, and FTP.

z/VM definitions

When an LPAR has been configured to benefit from hardware cryptography support, z/VM running in such an LPAR can use the hardware support for cryptographic operations to provide it to its guests. The way that z/VM provides this support is by gaining access to the adjunct processor (AP) queues to the guests. From a system implementation perspective, an AP of a Crypto Express2 feature is one of its internal cryptography engines (cryptography coprocessor units). Note that AP designates to the processor, while AP ID specifies the number associated with it.

To make use of the accessible hardware by z/VM and to provide it to the guests, note the following rules (see Figure 4-14 on page 93):

- Each AP can have up to 16 usage domains assigned to it.
- Each usage domain:
 - Has a separate set of master keys for secure key operation stored in the CEX3C.
 - Is associated with a separate AP queue.
- The AP queues reside in the hardware system areas (HSA) and provide access to an AP.
- An AP queue can be identified by the AP number and the usage domain index.
- The AP numbers are assigned to the Cryptographic Candidate List or Cryptographic Online List in the LPAR activation profile.
- Each LPAR is assigned at least one usage domain that applies to all of the APs configured to this LPAR.
- An AP can be shared among 16 LPARs.
- The combination of usage domain and AP must be unique among active LPARs.

According to these rules, a z/VM system can have up to 256 AP queues, which can be used by the z/VM guests.

⁵ Crypto Express3 Accelerator

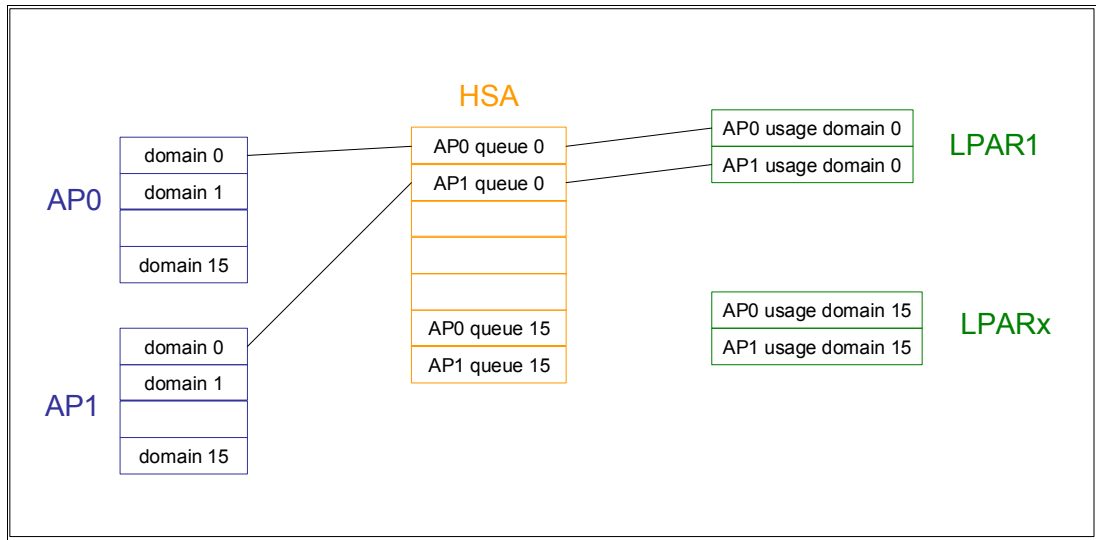


Figure 4-14 Mapping of AP queues to LPARs

In a z/VM environment it is expected that the LPAR running z/VM will have access to multiple AP queues. There are two ways that z/VM can provide access for the guests to the AP queues:

- Shared queue support
- Dedicated queue support

The shared queues support is the choice to provide for one or more Linux guests with hardware encryption support for clear key operation (for example, for SSL)⁶. With shared queue support, z/VM intercepts and simulates AP instructions for all shared-queue guests, and then issues the instructions on behalf of the guest. If there are multiple queues available, z/VM decides which AP queue is used under the cover. The dedicated queue support for a guest must be used if the guest requires secure key support and relies on stored encryption keys in the hardware coprocessors. For guests using dedicated-queue support, z/VM does not intercept and instead allows the guest to execute the AP instructions under SIE. In this case, no virtualization of AP queues is done.

Cryptographic software support: Linux

The System z platform, with its high availability, connectivity, scalability, and virtualization, provides an ideal platform on which to host Linux servers. Linux implementations have access to all of the cryptographic packages that any other Linux deployment might have, which typically includes many software cryptographic toolkits and products. What makes Linux distinct in this area is that it also has access to the System z hardware cryptography environment.

The Resource Access Control Facility

The Resource Access Control Facility (RACF) licensed program can satisfy the preferences of the user without compromising any of the concerns raised by security personnel. The RACF approach to data security is to provide an access control mechanism that offers effective user verification, resource authorization, and logging capabilities. RACF supports the concept of user accountability. It is flexible, has little noticeable effect on the majority of users, and little or no impact on an installation's current operation.

⁶ This also applies for z/OS and z/VSE guests, if only clear key support is necessary.

RACF controls access to and protects resources on both multiple virtual storage (z/OS) and virtual machine systems. For a software access control mechanism to work effectively, it must be able to first identify the person who is trying to gain access to the system and then verify that the user is really that person.

With RACF, you are responsible for protecting the system resources, such as minidisks, terminals, and shared file system (SFS) files and directories, and for issuing the authorities by which those resources are made available to users. RACF records your assignments in profiles stored in the RACF database. RACF then refers to the information in the profiles to decide whether a user should be permitted to access a system resource.

The ability to log information, such as attempted accesses to a resource, and to generate reports containing that information can prove useful to a resource owner and is very important to a smoothly functioning security system. Because RACF can identify and verify a user's user ID and recognize which resources the user can access, RACF can record the events where user-resource interaction has been attempted. This function records actual access activities or variances from the expected use of the system.

RACF has a number of logging and reporting functions that allow a resource owner to identify users who attempt to access the resource. In addition, you or your auditor can use these functions to log all detected successful and unsuccessful attempts to access the RACF database and RACF-protected resources. Logging all access attempts allows you to detect possible security exposures or threats. The logging and reporting functions are:

- ▶ **Logging:** RACF writes audit records in a file for detected, unauthorized attempts to enter the system. Optionally, RACF can also write records for authorized attempts or detected, unauthorized attempts to:
 - Access RACF-protected resources.
 - Issue RACF commands.
 - Modify profiles on the RACF database.
- ▶ **Sending messages:** RACF sends messages to the security console for detected, unauthorized attempts to enter the system and for detected, unauthorized attempts to access RACF-protected resources or modify profiles on the RACF database.
- ▶ **Keeping statistical information:** Optionally, RACF can keep selected statistical information, such as the date, time, and number of times that a user enters the system and the number of times a single user accesses a specific resource. This information can help the installation analyze and control its computer operations more effectively. In addition, to allow the installation to track and maintain control over its users and resources, RACF provides commands that enable the installation to list the contents of the profiles in the RACF database.

Today, RACF for z/VM is at RACF Feature Level 6.1 (with z/VM 6.1), offering the same features introduced with RACF 5.4 (on z/VM 5.4) without significant changes. A brief list of features includes:

- ▶ Mixed-case 8-character passwords.
- ▶ Mixed-case password phrases up to 100 characters, including blanks.
- ▶ No longer possible to reset password to default group names.
- ▶ Audit trail can be unloaded in XML format.
- ▶ Remote authorization and audit through z/VM new LDAP server and utilities.
- ▶ Password and password phrase enveloping and LDAP change logging of user and group profile updates.

LDAP

Today, people and businesses rely on networked computer systems to support distributed applications, which might interact with computers on the same local area network, within a corporate intranet, within extraneous linked up partners and suppliers, or anywhere on the worldwide Internet. To improve functionality and ease-of-use and to enable cost-effective administration of distributed applications, information about the services, resources, users, and other objects accessible from the applications must be organized in a clear and consistent manner. Much of this information can be shared among many applications, but it must also be protected to prevent unauthorized modification or the disclosure of private information.

Information describing the various users, applications, files, printers, and other resources accessible from a network is often collected in a special database that is sometimes called a directory. As the number of different networks and applications has grown, the number of specialized directories of information has also grown, resulting in islands of information that are difficult to share and manage. If all of this information could be maintained and accessed in a consistent and controlled manner, it would provide a focal point for integrating a distributed environment into a consistent and seamless system.

The Lightweight Directory Access Protocol (LDAP) is an open industry standard that has evolved to meet these needs. LDAP defines a standard method for accessing and updating information in a directory. LDAP has gained wide acceptance as the directory access method of the Internet and is therefore also becoming strategic within corporate intranets. It is being supported by a growing number of software vendors and is being incorporated into a growing number of applications.

z/VM V5.3 introduces a z/VM LDAP server and client. It is a subcomponent of TCP/IP. The z/VM LDAP server has been adapted from the IBM Tivoli Directory Server for z/OS (delivered in z/OS V1.8).

z/VM LDAP can help to simplify administration tasks when a Linux farm is installed on z/VM. The z/VM LDAP server helps administrators:

- ▶ Improve Linux logon security using RACF services to validate user and password.
- ▶ Reduce repetitive tasks, such as defining the same Linux user in multiple Linux images.
- ▶ Implement the RACF database sharing coupled with a z/VM LDAP server using native authentication permits to have a single point to administrate multiple z/VM and Linux servers.
- ▶ Gives a better business continuity solution when using multiple mainframes in conjunction with the hiperswap function.

LDAP upgrade and RACF password change logging

In order to help maintain cross-platform consistency, the z/VM LDAP server introduced in z/VM V5.3 has been upgraded to the function level of the z/OS V1.10 IBM Tivoli Directory Server for z/OS. The z/VM RACF Security Server feature has been enhanced to create LDAP change log entries in response to updates to RACF group and user profiles, including changes to user passwords and password phrases. This update enables password changes made on z/VM to be more securely propagated to other systems, including z/OS, using applications such as the IBM Tivoli Directory Integrator.

4.7.3 Important decisions with respect to security

With respect to security on System z the following decisions have to be made when adding a new workload to the environment:

- ▶ Will hardware crypto be required?
This is usually the case if the application requires encryption and the volume of transactions is high.
- ▶ Are external directories used for security policies and access information? If so, are these directories standardized, such as for example LDAP?
- ▶ Are specific authentication mechanisms required?

4.8 Transaction management

A workload may or may not include *transactions*. These transactions, or units of work (UOW) may have the requirement to be processed with a so-called 2-Phase Commit (2-PC) protocol. The 2-PC protocol ensures that multiple related updates to data in a unit of work are performed in a coordinated manner. If a failure would occur during the execution of the unit of work, updates already done would be backed out, so that data remains consistent. This failure could be as hard as a full sudden power down of the entire data center.

The concept of processing transactions this way is essential in certain workloads. Two examples that immediately come to mind are reservation systems, such as airline reservation systems, and payment systems. Processing transactions this way has been one of the cornerstones of the IBM mainframe for decades. It is also a key concept on IBM mainframes that ensuring reliability by means of transaction processing is a responsibility of middleware and the operating system. Customer Information Control System (CICS), Information Management System (IMS) and Transaction Processing Facility (TPF) are all heavily used transaction management solutions in many of the Fortune 1000 companies across the world. These transaction managers are fully integrated with database managers, such as DB2 for z/OS and file systems, such as Virtual Storage Access Method (VSAM).

Most of the WebSphere family of products also provide transaction management. Java Enterprise Edition (JEE) applications can be fully run transactionally inside WebSphere Application Server.

As noted earlier, IBM System z offers five different operating systems, of which z/TPF is an operating system by itself. The CICS and IMS transaction managers are only available on z/OS. On Linux for System z, transaction management is available through WebSphere and a number of ISV solutions.

4.8.1 Running programs in a transaction manager or not

Running programs in a certain transaction manager is not fully transparent, although JEE applications can practically run in any JEE application server and be moved from one JEE application server to another without real issues. For CICS, IMS and TPF, however, programs need to be specifically prepared and deployed. For example, a COBOL program that currently runs on a UNIX platform cannot be just moved to CICS on z/OS without conversion and recompilation. Programs currently running in an ISV transaction manager on a distributed platform can most likely be easily moved over to the same ISV transaction manager on Linux for System z, because they use the same APIs.

Based on the assessment of the current workload, it needs to be decided whether the workload should be run inside a transaction manager or not. Some workloads manage transactions themselves without the help of a transaction manager. After moving such a workload to System z (z/OS or Linux for System z), its transaction behavior would still be the same or even better, if underlying z/OS operating system features are properly exploited.

Generally, running a workload inside a transaction manager provides the best reliability, frees the developer from the burden of maintaining the transaction management logic and provides a common framework and APIs. If a mission-critical workload is not run in a transaction manager already, a workload migration could be the right moment to consider starting to use one!

4.8.2 Important decisions with respect to transaction management

With respect to transaction management on System z, the following decisions have to be made when adding a new workload to the environment:

- ▶ How mission-critical is the application and to what extent is 2-PC required to ensure reliability?
- ▶ Does the workload require an external transaction manager, such as CICS or BEA Tuxedo, or does it manage its own transactions?
- ▶ Is the workload a possible candidate for z/TPF?
- ▶ Is Linux for System z an option and are there ISV transaction managers available?
- ▶ If the workload is a JEE workload, should WebSphere Application Server on z/OS or WebSphere Application Server on Linux for System z be used?

4.9 Data management

In the following sections we provide an overview of the most important database management systems and file systems in z/OS and Linux on System z.

4.9.1 Database management systems

We discuss the following database management systems on System z.

z/OS:

- ▶ DB2 on z/OS
- ▶ Information Management System (IMS) on z/OS

Linux on System z:

- ▶ Oracle
- ▶ Informix®

DB2 on z/OS

Note the following facts about DB2 for z/OS: it is used by the top 59 banks in the world, as well as by 23 of the top 25 US retailers and by 9 of the top 10 global life and health insurance providers.

DB2 for z/OS delivers high performance, as indicated by the following facts:

- ▶ It delivered the largest banking benchmark ever at a large bank in Asia, a record 9,445 transactions per second.
- ▶ It processed 15,000 transactions per second, almost 300,000 SQL/sec for a large Asian bank benchmark.

- ▶ It supports the world's largest known peak database workload: 1.1 billion SQL statements per hour at UPS.
- ▶ It supports the world's largest known transaction processing database: 23.1 TB at a public records agency.

DB2 for z/OS is the leading enterprise data server, designed and tightly integrated with the IBM System z mainframe to use the strengths of System z, and to reduce TCO through process enhancements and productivity improvements for database administrators and application developers. New structures, such as the ability to make changes to data definitions without disrupting online performance, continue to enhance availability and scalability in DB2, and bottlenecks have been removed to ensure the position of DB2 as a performance leader.

The following examples show the deep synergy between DB2 on z/OS and the System z platform:

- ▶ Data sharing (to provide availability and scale out)
- ▶ Hardware data compression
- ▶ System z Integrated Information Processor (zIIP) specialty engines
- ▶ Unicode conversion
- ▶ Encrypted TCP/IP communication (SSL), encrypted data
- ▶ Cross-memory, memory protection keys
- ▶ Sorting
- ▶ Multi-core, large N-way
- ▶ 1 MB page size
- ▶ Decimal float arithmetic (z10)
- ▶ 64-bit addressing and large memory
- ▶ z/OS Workload Manager
- ▶ z/OS Security Server (RACF)
- ▶ z/OS RRS integrated commit coordinator

The following sections explain in more detail the four key strengths of DB2 on z/OS:

- ▶ DB2 data sharing
- ▶ Compression
- ▶ Partitioning
- ▶ Parallelism

DB2 data sharing

DB2 data sharing (Figure 4-15 on page 99) improves the availability of DB2 data, extends the processing capacity of the system, provides more flexible ways to configure the environment, and increases transaction rates. You are not required to change the SQL in your applications to use DB2 data sharing. All members of a data sharing group share the same DB2 catalog and directory, and all members must reside in the same Parallel Sysplex.

The DB2 data sharing design gives businesses the ability to add new DB2 subsystems into a data sharing group, or cluster, as the need arises and without disruption. It provides the ability to perform rolling upgrades of service or versions of the software stack without any application outage.

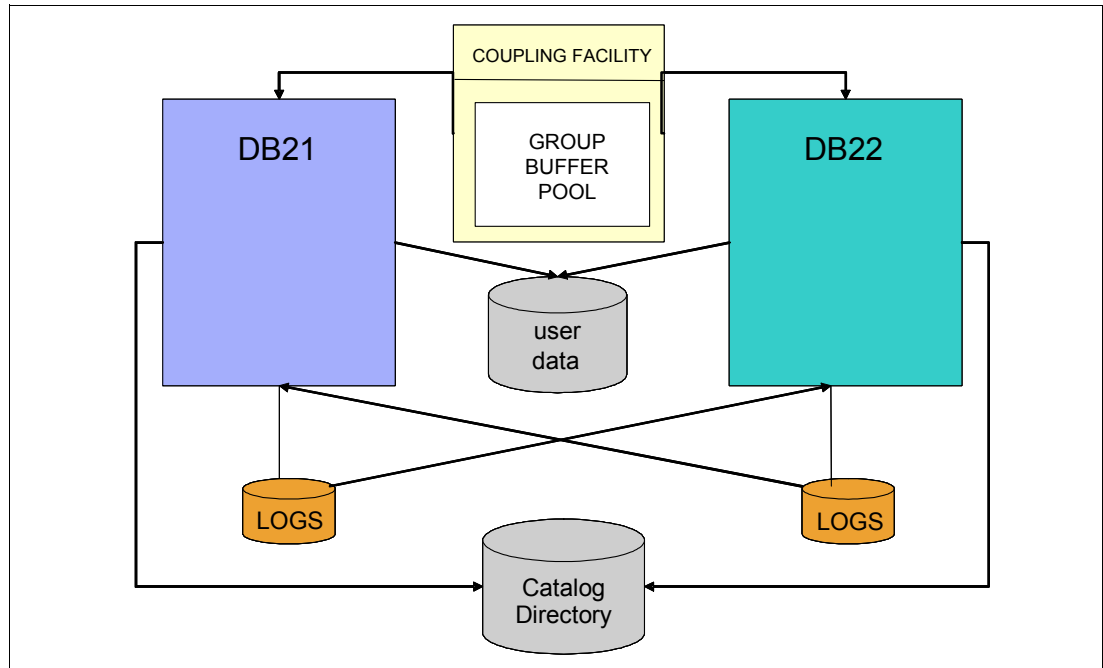


Figure 4-15 DB2 data sharing

With DB2 data sharing, you obtain the following benefits:

- Support for incremental growth

A Parallel Sysplex can grow incrementally, allowing you to add processing power in granular units and in a nondisruptive manner. The coupling technology of Parallel Sysplex, along with the additional processor power, results in more throughput for user applications. You no longer need to manage multiple copies of data, and all members of the data sharing group share a single DB2 catalog and directory.

- Workload balancing

DB2 data sharing provides workload balancing so that when the workload increases or you add a new member to the group, you do not need to distribute your data or rewrite your applications. DB2 data sharing is unlike the partitioned-data approach to parallelism (sometimes called shared-nothing architecture), in which a one-to-one relationship exists between a database management system (DBMS) and a segment of data. When you add a new DB2 subsystem onto another central processor complex (CPC) in a data sharing environment, applications can access the same data through the new member just as easily as through any of the existing members.

DB2 works closely with the Workload Manager (WLM) component of z/OS to ensure that incoming requests are optimally balanced across the members of a data sharing group. All members of the data sharing group have the same concurrent and direct read-write access to the data.

- Capacity when you need it

A data sharing configuration can handle peak work loads well (such as end-of-quarter processing). You can have data sharing members in reserve, bring them online to handle peak loads, and then stop them when the peak passes.

Compression

Counter-intuitively, compressing data can reduce elapsed time of most data warehouse-type queries. DB2 for z/OS compresses the rows on a page, so that each data page is full of

compressed rows. It uses the hardware instruction along with a data dictionary to provide the most efficient compression available. The compressed data can also be encrypted, thereby saving space and implementing security requirements at the same time.

With a rule of thumb of a 50% compression rate, a compressed page contains twice the rows that an uncompressed page contains. This means that each I/O retrieves twice as much compressed data as when the data is uncompressed. The data remains compressed in the buffer pool. This means that DB2 for z/OS can cache twice as much compressed data in its buffer pool as it retrieves if the data were uncompressed. Finally, when data is modified in a row that is compressed, the information logged about that data change is also compressed, thus reducing log volume.

Not all data on a compressed page is decompressed; only the row or rows that are needed by the application are decompressed. Combined with the use of a hardware instruction to perform the decompression, this decompression serves to limit the amount of additional processor resource that is needed to access compressed data.

The larger amount of data retrieved in each I/O is compounded with the DB2 9 for z/OS increased prefetch quantities. This provides significant elapsed time reductions for all types of sequential processes, including the typical business intelligence queries that use table scans and index range scans. This includes sequential processes for utility access, providing benefits in terms of faster reorganizations, faster unloads, and faster recovery.

Partitioning

In a data warehouse environment, the size, growth estimates, or both of certain tables, such as fact tables and history tables, pose a challenge for designing and administering the table space. They also present challenges in ensuring query performance, and loading and deleting data. Partitioned table spaces, as they have evolved with the implementation in DB2 9 for z/OS, help resolve most of these problems.

Your data can be partitioned as follows:

- ▶ **Range partitioning table space**

This is a universal table space or non-universal table space partition table space that uses ranges to define the various partitions. The number of partitions and the range of each partition is user-controlled.

- ▶ **Partitioned by growth table space**

This table space is always a universal table space. Additional partitions are added to a partitioned by growth universal table space as additional space is needed until a predefined maximum number of partitions is reached. Partitioned by growth universal table space ranges and new partitions added to a partitioned by growth universal table space are completely managed by DB2. A user has no control over the ranges used or when a partition is added.

- ▶ **Table partitioned**

Partitioning keys are defined within the table's description at CREATE TABLE time. Table partitioning does not use a partitioning index. It also separates clustering from partitioning.

- ▶ **Data partitioned secondary index**

This index is a type of partitioned index available in DB2 for z/OS Version 8 and later. A data partitioned secondary index has the same number of index partitions as the table space has partitions. It is defined with columns other than the columns that are used to define table-controlled partitions.

Parallelism

You can significantly reduce the response time for data- or processor-intensive queries by taking advantage of the ability of DB2 to initiate multiple parallel operations when it accesses data in a data warehouse environment.

Note the following types of parallelism:

- ▶ Query I/O parallelism manages concurrent I/O requests for a single query.
- ▶ Query CP parallelism enables true multitasking within a query.
- ▶ Sysplex query parallelism enables a large query across separate DB2 members in a data sharing group.
- ▶ DB2 can use parallel operations for processing the following types of operations:
 - Static and dynamic queries
 - Local and remote data access
 - Queries using single table scans and multitable joins
 - Access through an index, by table space scan or by list prefetch
 - Sort
- ▶ Utilities parallelism:
 - Unload and Load
 - Reorg
 - Copy

Why use DB2 on z/OS

The following reasons show why DB2 on z/OS is the best choice to implement the data warehouse:

- ▶ Perhaps the majority of source systems on z/OS are within IMS, VSAM, DB2, or sequential files, or there is a requirement for tight integration with existing resources and systems on the System z platform.
- ▶ The operational data store already exists on System z as several of the data warehouses and data marts.
- ▶ Existing skills and investments are on the System z platform.
- ▶ The company already maintains a System z-centric IT solution due to its favorable cost of ownership and comfort.
- ▶ The company is implementing an operational BI application with embedded analytics within applications. These types of applications can use the System z transaction scalability capabilities.
- ▶ There is a requirement for a true real-time operational data store:
 - Operational data is already on the System z platform.
 - Data must be virtually in sync with the operational data.
 - Availability, security, and resiliency requirements are high.
 - Auditable data warehouse requirements exist.
- ▶ Independent software vendor (ISV) packages, such as SAP and PeopleSoft on System z, offer both transactional (OLTP) and informational (warehouse and BI) systems. System z supports multiple images of SAP and other solutions, which simplifies support for these complex applications.

These packaged applications, which have tightly integrated components, have always made it desirable for the operational data and the warehouse to be housed in the same environment. Collocation reduces operational complexity, allowing for the reuse of skills and infrastructure in the form of processes, tools, and procedures.

- A desire already exists to consolidate distributed marts or data warehouses to an existing System z data serving platform. The client possibly has spare System z capacity.

Information Management System (IMS)

IMS is an IBM program product that provides transaction management and database management functions for large commercial application systems. It was originally introduced in 1968. There are two major parts to IMS: a transaction manager (TM) and a database manager (DB).

IMS TM is a message-based transaction processor that is designed to use the z/OS environment to your best advantage. IMS TM provides services to process messages received from the terminal network (input messages) and messages created by application programs (output messages). It also provides an underlying queueing mechanism for handling these messages.

IMS DB is a hierarchical database manager that provides an organization of business data with program and device independence. It has a built-in data share capability. It has been developed to provide an environment for applications that require very high levels of performance, throughput and availability, and to make maximum use of the facilities of the operating system and hardware on which it runs—z/OS and System z, respectively.

IMS TM and IMS DB can be purchased and used independently. Most companies using IMS have either a TM or a DB implementation, and some have both. The flexibility of z/OS and the IBM software products that run on z/OS allow combinations of products while fully guaranteeing the Quality of Services the platform stands for, such as:

- Using IMS TM for running application programs and using DB2 as the DBMS
- Using IMS DB as the DBMS and CICS for running application programs

Because we are talking about DBMSs in this section, we will not further concentrate on IMS TM, but continue to discuss IMS DB.

IMS DB

IMS DB is a comprehensive hierarchical DBMS solution including specific structures to organize and store databases, utilities and APIs. IMS databases are first designed and then defined using specific JCL. Databases consist of segments that are related to each other in a hierarchical manner. This means that segments can have parent or child segments. Application programs navigate through this structure in a high-performing way using specific APIs and the *DLI* interface. When written efficiently, programs can navigate through millions of segments in a matter of minutes.

Batch Messaging Program (BMP)

One very nice feature is that you can run batch programs on IMS databases during the OLTP window without disconnecting the databases. Batch Messaging Program (BMP) is a program that is started by submitting JCL. It performs bulk updates to the databases. BMPs are checkpoint restartable, which means that if a BMP abends or stops in the middle, you do not have to rerun the entire program from the beginning. When BMPs are planned to run during the OLTP window, it is important to do careful resource and workload planning so they do not take too many cycles and do not overload the IMS system.

SQL access to IMS databases

Even though IMS databases have a hierarchical structure and a specific API to access, they can also be accessed using SQL queries. IMS DB provides a layer that “translates” SQL to DLI calls.

Oracle in Linux on System z

The Oracle database server is a DBMS widely used in the marketplace. IBM System z provides a great advantage in terms of I/O bandwidth and high availability for this Oracle product. *Maximum Availability Architecture (MAA)* is Oracle's blueprint based on proven Oracle high-availability technologies and recommendations.

Oracle *Cluster Ready Services (CRS)* is a key component of Oracle MMA, wherein two or more entities of the operating system, known as *nodes*, work in an integrated manner sharing memory over the network. The storage area is controlled by a layer of software known as Oracle *Automatic Storage Management (ASM)*. With ASM it is possible that two or more nodes have access to reading from and writing to (RW) the same storage system.

Oracle *Real Application Cluster (RAC)* forms the integration between CRS and ASM. Figure 4-16 shows a simple implementation of Oracle RAC on IBM System z.

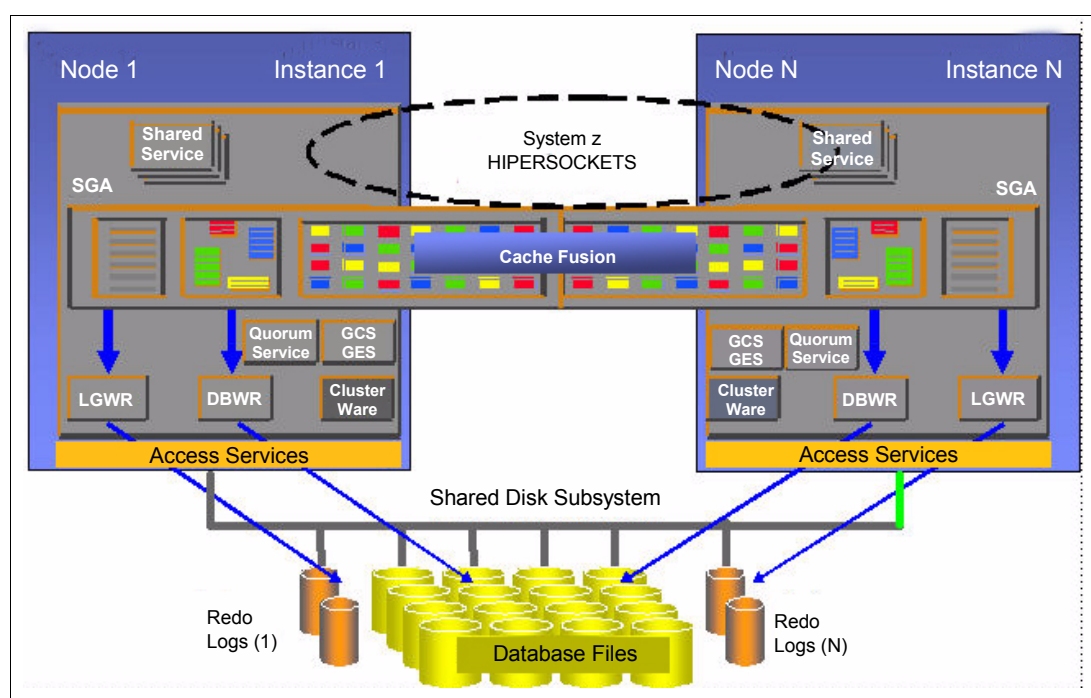


Figure 4-16 Oracle RAC on IBM System z

Cache Fusion is responsible for the integration between the RAC nodes, and leverages IBM System z HiperSockets. In this model of communication only pointers to data memory are transmitted from one node to another. Cache Fusion is used to establish a “private network” environment of Oracle CRS. More details about HiperSockets can be found in 4.5.3, “System z network connectivity” on page 69.

Oracle CRS also has another communication service called *public network*, which is the method of client access to the database via the network. This service is configured using the Virtual IP Address (VIPA) methodology, ensuring access through the network to any one of the nodes, even in case some are not available. VIPA makes sure that access is transferred to the node that is available (online).

The memory of the Oracle database server is called *System Global Area (SGA)*. With Oracle CRS the SGA is shared across the nodes with Cache Fusion, guaranteeing coordination of data updates in different nodes in such a way that whenever a client queries data it always receives the current version.

Another example of high availability using Oracle products is a model in which a standby database is used. This solution is called *Oracle DataGuard*. The standby database is separate from Oracle CRS and is passive only. If the Oracle CRS fails, the Oracle DataGuard database takes over and continues to respond to client connections. This database needs a separate storage subsystem and a separate network infrastructure. An example of this environment is shown on Figure 4-17.

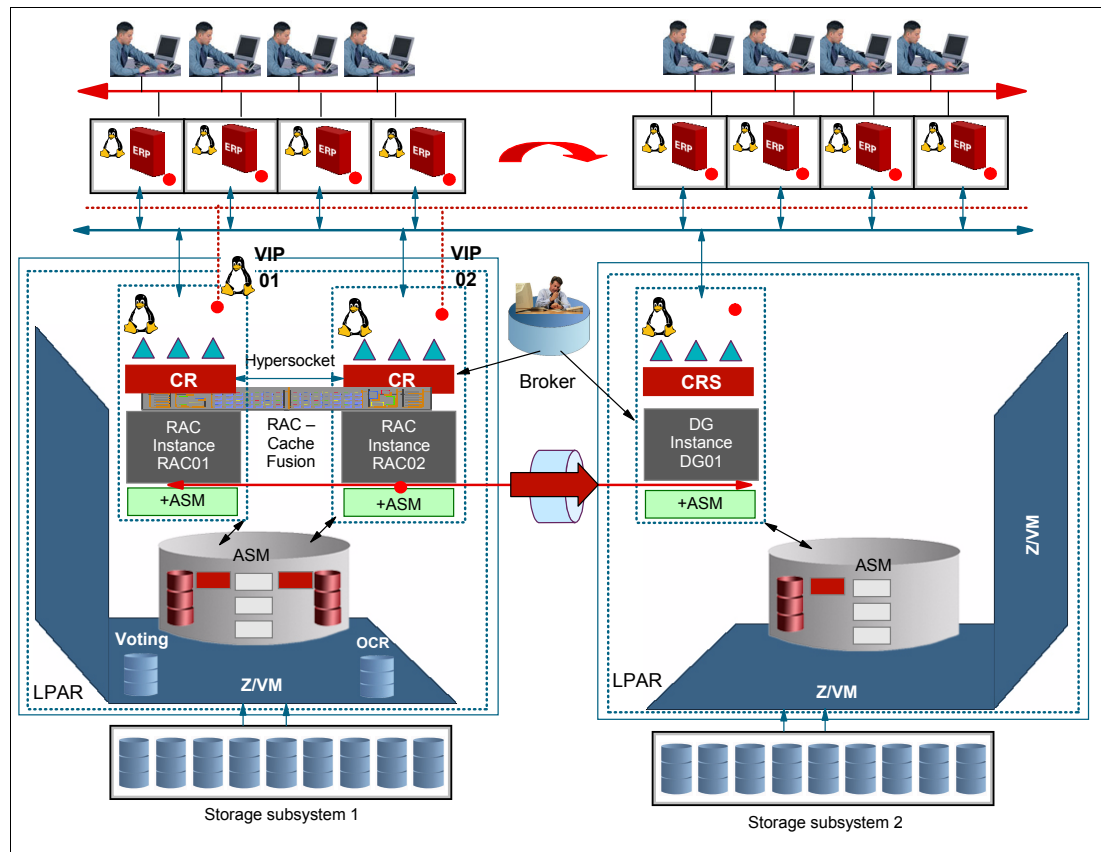


Figure 4-17 Oracle CRS with Oracle DataGuard environment on IBM System z

Another aspect of Oracle MAA is the *Oracle Grid Control*. This solution controls all Oracle products in client environments from a single point of control. The control extends to database and application servers with functions for monitoring, configuration, management, security, historical statistical, events, and alerts.

Oracle is a best fit solution to run on IBM System z, as it exploits much of the I/O subsystem and this decreases the processor utilization of required IFLs. This means you need relatively fewer processors to consolidate more workload.

Other DBMSs in Linux on System z

Another DBMS with high availability options is an Open Source solution called *PostgreSQL* (PGSQL).

PGSQL is used in many web applications on the Internet and has the capability to run in high availability mode. There are different modes for HA based on either synchronous or asynchronous synchronization.

In the case of synchronous synchronization, a data-modifying transaction is not considered committed until all servers have committed the transaction; this guarantees that a failover will

not lose any data and that all load-balanced servers will return consistent results no matter which server is queried. In asynchronous mode, some delay between the time of a commit and its propagation to the other servers is allowed, opening the possibility that some transactions might be lost in the switch to a backup server, and that load-balanced servers might return slightly stale results. Asynchronous communication is used when synchronous would be too slow.

Performance must be considered in making the right choice. There is usually a trade-off between functionality and performance. For example, a fully synchronous solution over a slow network might cut performance by more than half, while an asynchronous one might have a minimal performance impact.

The most common implementation of HA with PGSQL is a mode that considers an active/passive environment, with one or more *standby servers* ready to take over operations if the primary server fails. This capability is widely referred to as *warm standby* or *log shipping*. In this mode the primary and standby server work together to provide this capability, though the servers are only loosely coupled. The primary server operates in continuous archiving mode, while each standby server operates in continuous recovery mode, reading the *write-ahead log (WAL)* files from the primary. No changes to the database tables are required to enable this capability, so it offers low administration overhead in comparison with some other replication approaches. This configuration also has relatively low performance impact on the primary server.

The standby server is not available for access, since it is continually performing recovery processing. Recovery performance is sufficiently good that the standby will typically be only moments away from full availability once it has been activated. As a result, we refer to this capability as a warm standby configuration that offers high availability. Restoring a server from an archived base backup and rollforward will take considerably longer, so that technique only offers a solution for disaster recovery, not high availability.

4.9.2 Highly available file systems

In the following sections we discuss some file systems that can be used on IBM System z.

z/OS

In z/OS there are two types of file systems:

- ▶ Traditional MVS data sets, which are used to store data on System z since the very beginnings of the predecessors of the z/OS operating system.
- ▶ File systems that are used to store UNIX directories and its contents. These directories are very similar in usage to directories in other UNIX flavors, but are physically stored in System z file systems.

MVS data sets

in z/OS, the default mechanism for storing information on disk or tape is by using *MVS data sets*. There are a number of different types of data sets, each suitable for specific tasks. Depending on the type, data is organized and accessed in a certain manner. For example, an essential difference is whether you expect to read through data sequentially or whether you wish to access the data by using a key. Also, data sets may contain such a large volume of data that indexing becomes necessary. We now discuss the three main types of data sets and how they could play a role in your migrated workload.

Sequential In a *sequential* data set, records are data items that are stored consecutively. To retrieve the tenth item in the data set, for example, the system must first pass the preceding nine items. Data items that must all be used in sequence,

like the alphabetical list of names in a classroom roster, are best stored in a sequential data set.

Partitioned A *partitioned* data set or PDS consists of a directory and members. The directory holds the address of each member and thus makes it possible for programs or the operating system to access each member directly. Each member, however, consists of sequentially stored records. Partitioned data sets are often called libraries. Programs are stored as members of partitioned data sets. Generally, the operating system loads the members of a PDS into storage sequentially, but it can access members directly when selecting a program for execution.

VSAM In a *Virtual Storage Access Method (VSAM)* key sequenced data set (KSDS), records are data items that are stored with control information (keys) so that the system can retrieve an item without searching all preceding items in the data set. VSAM KSDS data sets are ideal for data items that are used frequently and in an unpredictable order.

Data sets can be shared between multiple LPARs within a Parallel Sysplex. Optionally, data sets can be kept in the Coupling Facility (CF)⁷, which is typically the highest available component in the solution.

zFS

Since the introduction of OS/390, the predecessor of the z/OS operating system, it is possible to run UNIX-compliant applications on System z. For this purpose a layer has been introduced called *UNIX System Services*. This layer provides a UNIX-95 compliant interface layer for applications as well as a user interface environment. Unique in this aspect is that a user can either work with the UNIX environment in a telnet fashion or by using ISPF panels, depending on the background of the user. A telnet user would not notice the difference between working with UNIX System Services on System z and any other UNIX environment. The commands are generally the same.

On System z, the logical UNIX directory structure is mapped onto a physical file system (PFS). On z/OS several options exist, but the recommended physical file system to use is the *z/OS Distributed File Service zSeries File System (zFS)*. Figure 4-18 on page 107 shows an overview of UNIX applications using files on z/OS.

⁷ Refer to "Parallel Sysplex" on page 77.

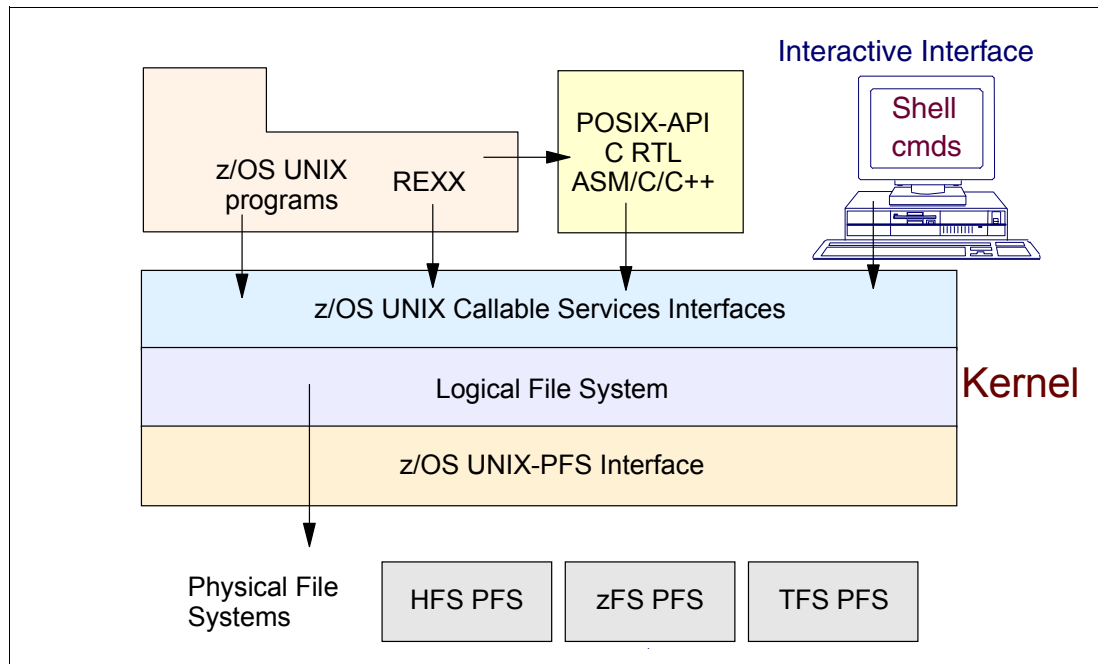


Figure 4-18 UNIX file systems in z/OS

In case UNIX directories and files are used by the application to read and write data, then these need to be part of the scope of the high availability and disaster recovery solution, like a DBMS would be. zFS file systems can be configured and shared between multiple LPARs within a Parallel Sysplex environment.

4.9.3 Important decisions with respect to data management

With respect to data management on System z, the following decisions have to be made when adding a new workload to the environment:

- ▶ Where is the data currently stored, is it a relational database, an indexed file system, or just flat files?
- ▶ What is the access method of the data?
- ▶ Are the data access routines separate components in the application that can be easily changed or replaced?
- ▶ If the application is coming from an ASCII-based platform, is conversion to EBCDIC required or can everything be kept in ASCII?
- ▶ Is a hybrid of Linux and z/OS a possibility?

The benefit of this model is that Linux can be exploited to provide application scalability and z/OS to provide extreme reliability and availability of databases.

4.10 Programming environment

On System z, both in z/OS and Linux for System z, a variety of programming languages is supported. We can group programming languages into the following categories:

- ▶ Platform-transparent “open” languages, such as Java and PHP. Both are supported on z/OS and Linux for System z.

- ▶ Common programming languages that are in wide use across multiple platforms. COBOL and C/C++ are the most common ones. COBOL, PL/I, Fortran and C/C++ are available on both z/OS and Linux for System z.
- ▶ Programming languages that are proprietary to a certain platform, such as z/OS Assembler and Microsoft .Net. It is with this type of programming language where most of the issues arise in a workload migration or conversion.
- ▶ Scripting languages, such as Perl, CGI, REXX and Ant. Most modern scripting languages are available on multiple platforms and typically run under Windows or UNIX. So, in the case of z/OS, a script is typically run in UNIX System Services. On Linux for System z, usage of scripting languages is as common as in any other Linux or UNIX environment.

In the following sections we discuss more details about these programming languages and the tools available.

4.10.1 Java on System z

Java in general, J2SE and the JEE programming model are fully supported on both Linux for System z and z/OS. A Java application always runs inside a *Java Virtual Machine (JVM)*. On z/OS, a JVM can be started stand-alone by a shell script, from the command line, JCL, or it can be started by WebSphere Application Server, CICS, DB2 or IMS. On Linux for System z, a JVM can be started stand-alone by a shell script, from the command line, or by any runtime product using a JVM, such as WebSphere Application Server.

On z/OS, Java classes are stored inside directories in the UNIX System Services environment. On Linux for System z they are stored inside directories. A developer can access these directories transparently from the workstation with any UNIX- or Linux-capable emulator or protocol.

Native extensions on z/OS

On z/OS, Java has a few z/OS-specific extensions for MVS data set access (JZOS Toolkit) and security functions. These extensions are not available in Java applications running on non-z/OS platforms.

Codepage considerations with Java on z/OS

On z/OS, the default codepage is 1037 within UNIX System Services and 037 in the MVS environment. Both are EBCDIC codepages. However, the Java classes running within the JVM on z/OS run in a Unicode environment, which is ASCII-based. When a Unicode Java class accesses an EBCDIC file or data set using standard Java classes (for example, java.io in UNIX System Services or the JZOS Toolkit in the MVS environment), automatic codepage conversion takes place on the data written to or read from the file.

Also, when a Unicode Java class accesses an EBCDIC database (for example, DB2) using standard Java Database Connectivity (JDBC) or a CICS or IMS transaction using CICS Transaction Gateway or IMS Connect, respectively, codepage conversion is done by the connector frameworks. It is important to be aware that native access from Java classes to EBCDIC resources, such as a TCP/IP request to a CICS region, will not provide standard codepage conversion.

4.10.2 COBOL and PL/I

Both COBOL and PL/I have matured on z/OS. The latest versions of COBOL and PL/I provide support for XML parsing and interoperability with Java. COBOL and PL/I can be used as

stand-alone programs and within transaction environments such as CICS, IMS and DB2 stored procedures.

On z/OS, COBOL and PL/I programs always have to be compiled and linked on the runtime platform, which is typically done with customized JCL. Most companies with System z already have a highly sophisticated environment for code build, deployment and testing.

COBOL and PL/I are also available on Linux for System z.

Codepages

COBOL and PL/I programs used on other platforms than z/OS may have to be converted to run on z/OS.

4.10.3 C/C++

C/C++ is supported on Linux for System z, z/VM, and z/OS.

Linux for System z

The GNU Compiler Collection (GCC) is available for both C and C++ and supports the latest IBM hardware, the Enterprise z196.

z/VM

The XL C/C++ compiler is available on z/VM and shares the same technology found on AIX, z/OS, BlueGene and Linux on POWER® systems. The z/VM C/C++ compiler includes advanced compiler optimization technology such as the high order transformation optimization (HOT), and also support for industry standards, thus simplifying any porting effort. Since z/VM is a 31 bit architecture, only ARCH(4) is supported.

z/OS

The z/OS XL C/C++ compiler is available not only as an MVS application, but also as a Unix System Services (USS) one, and supports both JCL style options (for example “OPT(3)”) as well as UNIX style command line options (for example “-O3”). The z/OS XL C/C++ compiler supports an array of powerful optimizations, including the high order transformation (HOT), profile directed feedback (PDF) and inter-procedural optimization (IPA).

In addition to advanced optimization, the z/OS XL C/C++ compiler can aggressively exploit the hardware features available on System z machines. IEEE, hexadecimal and decimal floating point arithmetic is all supported in System z hardware, and the z/OS XL C/C++ compiler allows the programmer to exploit these instructions. As on Linux for System z, the latest instructions available on Enterprise z196 machines and both 31-bit and 64-bit addressing modes are available.

In addition to the performance related features the z/OS XL C/C++ compiler, like the one on z/VM, shares the compiler technology found on other IBM platforms, and supports industry standards such as the C++ Standard Template Library, thus greatly easing any porting effort. Not only will the source code be easily portable, but any makefiles as well, since many of the UNIX style options are the same.

Metal C

Normally, C/C++ programs require the Language Environment® (LE) to provide not only the C/C++ runtime support, but also more basic functionality such as the establishment of an execution context, including the heap and dynamic storage areas. Normally these dependencies prevent you from running C (or C++) generated code that runs in an environment where LE does not exist.

The XL C METAL compiler option generates code that does not require access to the Language Environment support at run time. Instead, the METAL option provides C-language extensions that allow you to specify assembly statements that call system services directly. Using these language extensions, you can provide almost any assembly macro, and your own function prologs and epilogs, to be embedded in the generated high level assembly (HLASM) source file.

Because a freestanding program does not depend on any supplied runtime environment, it must obtain the system services that it needs by calling assembler services directly. Fortunately, you do not always have to provide your own libraries, as the XL C compiler supplies a subset of the XL C runtime library functions for use with Metal C. This subset includes commonly used basic functions such as `malloc()`.

Scripting languages

The exact requirements for using a certain scripting language are usually determined by the workload or the middleware being used to run the workload. Scripting languages as we know them in the distributed world are not common in the traditional z/OS environment, and instead JCL, CLIST or REXX are used for this purpose. Within the UNIX System Services under z/OS, shell scripts can be used in the same way as they are used in other UNIX environments. In Linux for System z, the usage of scripting languages is the same as in other Linux or UNIX environments.

In web server environments, scripting languages such as Perl and CGI can be used for simple programming. For more serious programs Java is recommended.

Middleware products on System z may provide their own specific scripting language support. A good example of this is the usage of Jacl or Jython in the **wsadmin** command line interface of WebSphere Application Server.

It is always important to keep in mind that the purpose of scripting is to facilitate an application program, but it is not intended to be used for application programming itself.

The IBM Language Environment (LE) on z/OS

The IBM Language Environment (LE) can be seen as one of the most unique concepts of z/OS. LE provides a common programming environment in which modules written in different programming languages can be combined into one load module. COBOL, PL/I, VS Fortran and C/C++ can all be used as part of LE.

Language Environment combines essential and commonly used runtime services, such as routines for runtime message handling, condition handling, storage management, date and time services, and math functions, and makes them available through a set of interfaces that are consistent across programming languages. With Language Environment, you can use one runtime environment for your applications, regardless of the application's programming language or system resource needs because most system dependencies have been removed.

Figure 4-19 on page 111 shows a high-level overview of LE on z/OS.

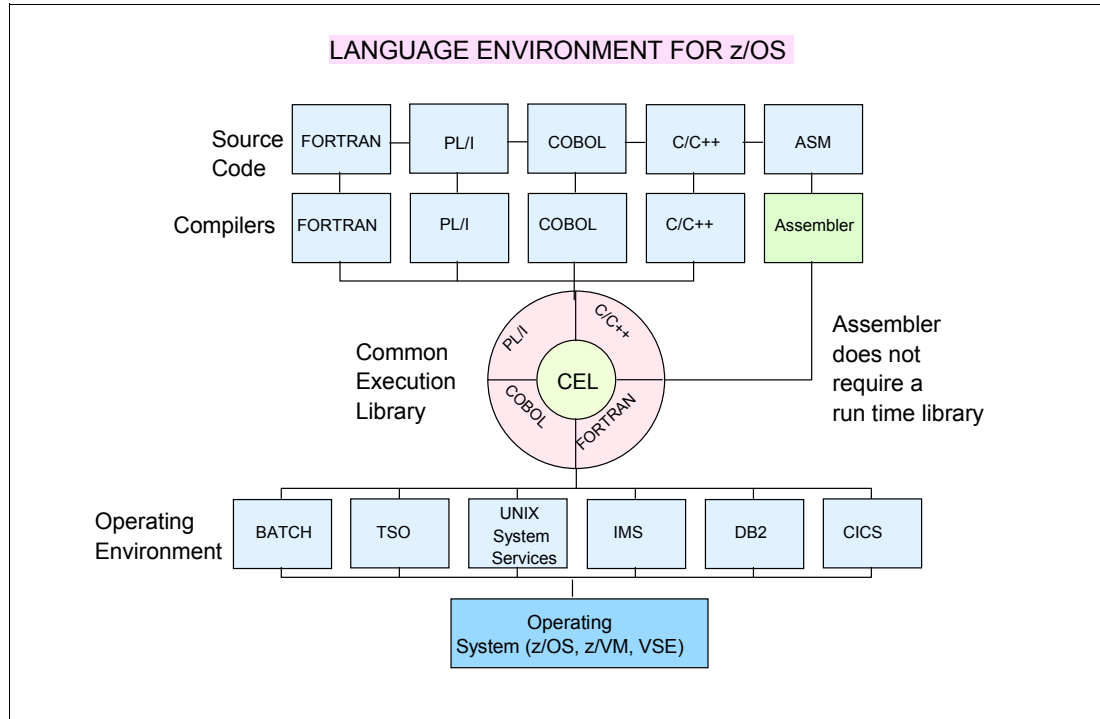


Figure 4-19 Language Environment (LE) on z/OS

4.10.4 Application development tooling

Application programs can be created and maintained on System z, without necessarily using additional workstation tools. It is proven, though, that workstation-based application development tools or “IDEs” (Integrated Development Environment) increase the productivity of the developer and the code quality significantly. Nevertheless, many System z application developers still use native on-board tools to do the job.

Under the different operating systems supported on System z, different native tools are provided for application development and maintenance tasks. For both environments, though, it is recommended to consider workstation-based application lifecycle management tooling. We now discuss the tooling options for Linux for System z and z/OS.

Linux for System z

The Linux for System z environment provides the same tools for editing and terminal emulation as you find in any other Linux environment. And most of these tools are the same as the ones being used in UNIX environments. Bottomline: application development for Linux on System z means hardly any change to application developers who already develop for Linux on any other platform or for UNIX.

It is recommended, though, to consider a workstation-based tool or workbench. There are workbenches on the market for any programming language. If you already use a workstation-based tool or workbench for application development, you can most probably continue using it, even though Linux for System z may mean a new runtime platform for you. This is especially the case for Java, because there is no compilation on the runtime platform involved. For programming languages that require compilation on the runtime platform, you need to verify whether and how your workstation-based tool or workbench supports remote compilation and link-edit on Linux for System z.

z/OS

z/OS provides its own environment for creating and maintaining application programs. *Time Sharing Option (TSO)* and *Interactive System Productivity Facility (ISPF)* are the two most important features under z/OS for the application developer. A user logged on to z/OS has a TSO session and is assigned resources. ISPF is a menu-based environment in which files can be created, edited, copied, and so on. Most z/OS users have their own ISPF macros to further facilitate editing.

Assembler, COBOL, PL/I, Fortran and C/C++ modules are compiled and linked using *Job Control Language (JCL)*, resulting in executable modules. Java on z/OS runs inside a Java Virtual Machine (JVM) like on any other platform.

Workstation-based tooling can be used for developing and maintaining application programs running on z/OS. IBM's flagship product for this purpose is *Rational® Developer for System z (RDz)*, which provides a true server-transparent experience. With proper workstation-based tooling, application developers hardly need any z/OS skills.

4.10.5 Important decisions with respect to programming environment

With respect to the programming environment on System z the following decisions have to be made when adding a new workload to the environment:

- ▶ Does the required programming environment dictate the choice of Linux for System z or z/OS?
- ▶ If the workload is written in a language not available on the target O/S, such as .Net, to what language will the workload need to be converted?
- ▶ If the workload is written in a mainstream programming language, such as COBOL or C/C++, will it need to be moved under a transaction manager such as CICS or IMS, or will it run stand-alone?

4.11 Integration

Integration between workloads takes place at different layers and in different styles.

4.11.1 Integration layers

With respect to layers, we could think about the 7-layer *Open System Interconnection (OSI) reference model* to classify integration layers. The top layer is the application layer and the bottom layer is the physical layer. At each layer, there are certain protocols that can be used to fulfill the objective of that layer.

When a workload is migrated to System z, it has to “fit” into its new environment at all layers. In the era of open systems, and System z being fully open, we are not that worried anymore these days about the protocols used by the workload at the bottom layers (up to layer 4), and protocol transparency is provided by hardware and the operating system. As an example, a workload currently using TCP at the Transport layer and IP at the Network layer will be able to use these same protocols on System z too.

At the top layers (layer 5 through 7), protocol transparency is provided by middleware. Which protocols can actually be used depends on the combination of the middleware and programming model.

With respect to integration, it is important to distinguish between:

- ▶ Workloads that run within common middleware, such as a JEE application server.

As a general rule, this type of workload is the easiest to migrate, because the combination of hardware, operating system and middleware will practically ensure integration at all layers and support the same protocols.

As an example, a workload running in WebSphere Application Server on a UNIX machine can be moved to WebSphere Application Server on z/OS or Linux for System z without having to worry about integration at any of the seven OSI reference model layers.

- ▶ Workloads that run in non-common middleware.

This type of workload may need to be migrated to another middleware on System z and therefore may or may not support the same integration protocols at the higher layers (layer 5 through 7) of the OSI reference model.

The choice of the transaction manager or application server on System z may depend on the integration protocols being used for these layers.

- ▶ Stand-alone workloads, not running within middleware.

This type of workload runs on top of the operating system without using middleware, such as a transaction manager or an application server. It is important to analyze (refer to Chapter 3, “Target architectures for availability” on page 15) which protocols, if any, are being used in the workload, especially the ones at the higher layers (layer 5 through 7) in the OSI reference model.

4.11.2 Integration styles

In the previous section we discussed the role of integration layers and protocols supported in each layer. To keep things easy, we now focus only on the application layer and distinguish between three integration styles:

- ▶ Synchronous
- ▶ Asynchronous
- ▶ Service-oriented

Synchronous

Synchronous integration, also known as Remote Procedure Call (RPC) style integration simply means that the calling program is waiting for the response to come back and then resumes processing. Synchronous communication can be achieved without using middleware. For example, a program can send a TCP/IP request to a listener, the listener processes the request, sends a response back and the calling program resumes processing again upon receipt of the response. Other techniques of synchronous integration on z/OS are Advanced Program-to-Program Communication/Common Programming Interface for Communications (APPC/CPI-C) and CICS and IMS transaction processing environments.

Synchronous integration can be a necessity in workloads where fast response times and transactional integrity are required, such as airline reservation systems, banking systems and stock trading systems.

Asynchronous

With asynchronous integration the calling program does not necessarily wait for the response to come back and the called program is not necessarily required to respond back immediately. In this scenario queues are needed to store the requests for a shorter or longer

period of time. Asynchronous integration on System z is supported by WebSphere MQ and WebSphere MQ-based middleware solutions.

When using asynchronous integration, multiple Units of Work (UOWs) are connected to each other.

Service-oriented

Service-oriented integration is a further advancement of asynchronous integration and is enabled by middleware, such as transaction servers (CICS and IMS), database servers (DB2), application servers (WebSphere Application Server) and integration middleware, such as WebSphere Message Broker (WMB).

In a service-oriented architecture, business functions are accessible as a *service*. The key concept of a service is that it can be accessed asynchronously from anywhere in the application environment.

Without the appropriate middleware, service-oriented integration cannot be implemented. All IBM middleware on z/OS and Linux for System z support service-oriented integration. Additionally, on Linux for System z there are also many ISV middleware solutions supporting service-oriented integration.

4.12 Systems management

IT systems management functions have evolved over the years in response to changes, demands and complexities of the IT infrastructure and services in order to maintain or improve high availability. In this section, we discuss the following mainframe IT systems management disciplines:

- ▶ Availability and performance management
- ▶ Capacity management
- ▶ Security management
- ▶ Service level management
- ▶ Problem and change management
- ▶ Configuration management
- ▶ Release management
- ▶ Storage management
- ▶ Service continuity management
- ▶ Incident management
- ▶ Asset and financial management

4.12.1 Availability and performance management

The purpose of *availability and performance management* is to ensure that the availability and performance of the IT infrastructure and services meet or exceed the requirements of the business. In many cases these requirements are defined via documented and/or negotiated SLAs or SLOs.

Availability management involves the planning for current and long-term service availability and the tracking and reporting of the results. This can include System z hardware, systems

software (z/OS, z/VM or Linux on System z), subsystems (CICS, IMS, DB2, and so on) as well as application availability.

Performance management involves the monitoring, measurement and analysis, tuning, tracking and reporting of performance from a system, subsystem and/or application perspective.

Figure 4-20 shows an example of a System z environment and the various monitoring points.

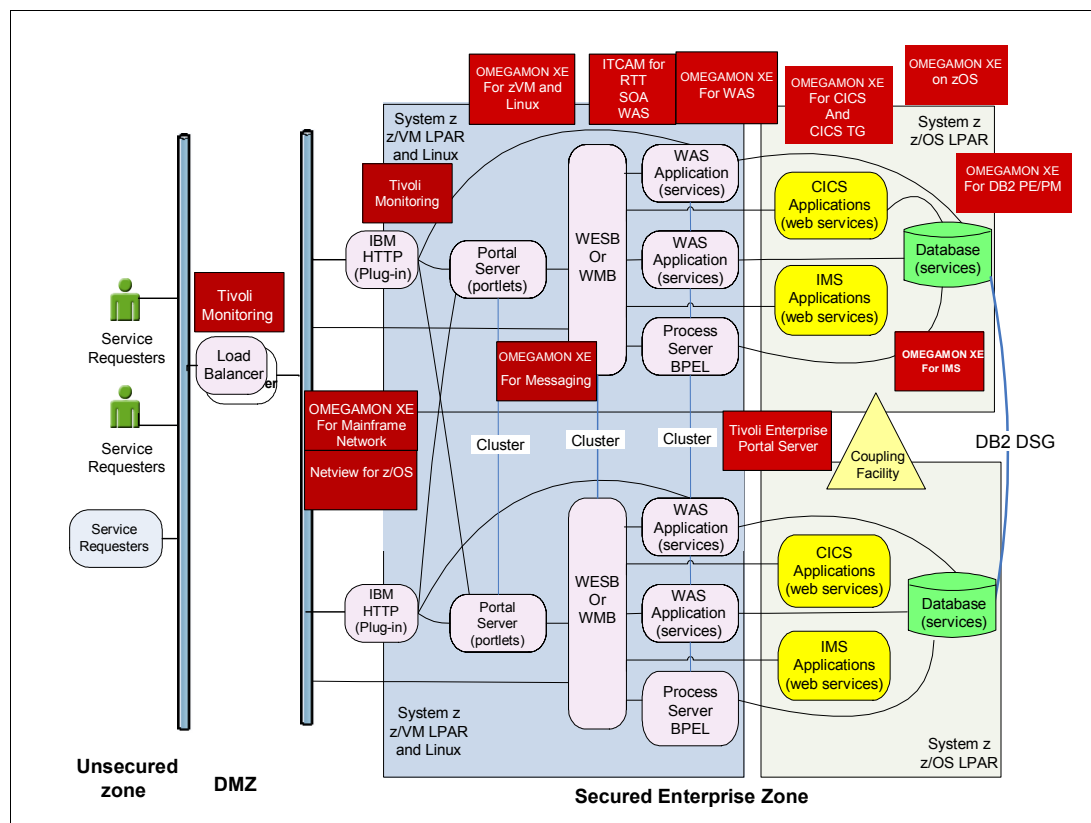


Figure 4-20 Sample environment with Tivoli availability and performance monitoring

Availability management

An efficient and effective availability management process is extremely important for maintaining a high availability environment. In developing the availability management process, the following key decisions need to be made:

- ▶ Determine availability requirements of the infrastructure components (hardware, LPARs, software and key business applications). These should address the business user and IT stakeholder requirements, which would then be translated into quantifiable availability terms.
- ▶ Define the availability targets. This involves the negotiation of achievable availability targets based on business needs and IT capabilities and resources available.
- ▶ Identify the monitoring, analysis, and reporting processes. These should define continuous monitoring and analysis of operational results, the reporting tools, and the reports to be utilized. It also involves the establishment of availability trends, the identification of issues of unavailability and the investigation of underlying causes of problems of missing availability SLAs.

As new applications are implemented and the changes are made to the infrastructure, it is important to update the availability management processes to keep them current.

System automation solution

To maintain a high availability environment, we need to minimize downtime. This usually requires the implementation of a robust system automation solution. A policy or script-based automation software product (when implemented properly) can provide easier management of a complex infrastructure and reduce downtime with improved availability and operational continuity. Key criteria for a system automation solution are:

- ▶ Provide self-detecting and auto-correcting capabilities
- ▶ React to events that threaten infrastructure performance and availability
- ▶ Eliminate error-prone manual intervention, which will improve availability and speed recovery times
- ▶ Automate time- or event-driven changes to the infrastructure

IBM Tivoli System Automation for z/OS (SA for z/OS) is a common host automation solution and is based on the IBM Tivoli Netview for z/OS product. It can ensure high availability for critical applications and System z infrastructure and services, including the Linux on System z environment.

In particular, SA for z/OS can provide the following benefits:

- ▶ Automate processor and system operations related to channels, ESCON® and FICON® directors, devices and JES-managed initiators based on time-driven scripts
- ▶ Provide console automation management for z/OS, z/VM and Linux on System z systems, including automated startups (IPLs) and shutdowns, management of system messages, and triggering of system responses based on event-driven scripts.
- ▶ Provide management and automation covering subsystems such as CICS, IMS and DB2 as well as WebSphere and MQ series environments. Subsystem tasks can be brought up or taken down automatically based on time-driven scripts. In addition, messages generated by these subsystems and applications can be automatically managed.
- ▶ Provide advanced IBM Geographically Dispersed Parallel Sysplex (GDPS) capabilities and integration, which can minimize downtime in failover situations
- ▶ Integrate with IBM Tivoli Workload Scheduler (TWS), allowing the TWS user to issue SA/Netview commands

Workload scheduler solution

In order to improve availability, throughput and utilization of resources, a workload scheduler solution should also be implemented. Key features of a workload scheduler solution should include the following:

- ▶ Automate, plan and control production workload processing
- ▶ Perform dynamic real-time workload automation in addition to event, calendar and time-based scheduling
- ▶ Automate processing of heterogeneous work in order to minimize idle time, improve throughput and utilize available resources
- ▶ Trigger batch job and jobstream execution based on real-time events, and job dependencies
- ▶ Monitor and manage workloads by exception and provide production runtime reports and exception reports when unusual conditions occur in the infrastructure or in the batch scheduling activity itself

- Provide job and workload management with real-time alerts and user notification as necessary, self monitoring, self-healing, automated recovery capabilities and the ability to support user-defined event rules for batch job scheduling

IBM Tivoli Workload Scheduler (TWS) for z/OS is a job and workload management automation tool that integrates with other systems management products. It provides the above capabilities as well as the following features and benefits:

- Monitors System z with the *IBM Tivoli Dynamic Workload Console (TDWC)*, the web-based GUI feature that is shipped as part of TWS for z/OS.
- Performs critical path analysis on workloads.
- Monitors System z resources with its integration with the IBM Tivoli System Automation for z/OS product.
- Interfaces with the *Tivoli Enterprise Portal (TEP)* to monitor job events and status. See the TEPs reference in “Performance Management”.

Performance Management

Performance monitoring, tracking and reporting can be done at the system or subsystem level either after the fact or in real time. In any case, once performance measurement data is obtained, it can be analyzed and compared to pre-established *Key Performance Indicators (KPIs)* to ensure efficient use of resources. Subsequently, tuning can take place to improve performance if warranted or to meet service level agreement criteria.

Improving performance could involve improving response time for transaction or database applications, decreasing job turnaround for batch jobs, or increasing throughput at the system level. In general, performance tuning could involve changing parameters at the system (z/OS) or subsystem (CICS, IMS or DB2) level to remove bottlenecks, adding hardware resources (such as memory, channels, faster I/O devices, CPs or specialty engines), or possibly even modifying code at the transaction or program level (application tuning).

From an “after the fact” monitoring and analysis perspective, performance measurement data can be collected by *System Management Facility (SMF)* or by *Resource Measurement Facility™ (RMF™)* and stored in the *IBM Tivoli Decision Support (TDS)* system. QMF™ queries can be developed to track data against KPIs, produce reports and show trending of performance metrics. These reports can then be automated to run on a regular basis (such as daily) by TWS.

For real-time monitoring, data can be collected by performance monitors at the infrastructure and operating system level as well as at the subsystem level. Examples of real time monitors that monitor the infrastructure include the following products:

- IBM Tivoli Monitoring for Linux on System z agent
- IBM Tivoli Omegamon XE on z/OS
- IBM Tivoli Omegamon XE on z/VM and Linux
- IBM Tivoli NetView® for z/OS
- IBM Tivoli Omegamon XE for Mainframe Networks

Examples of real-time monitors that monitor subsystems and WebSphere applications include the following:

- IBM Tivoli Omegamon XE for CICS on z/OS
- IBM Tivoli Omegamon XE for CICSSTG on z/OS
- IBM Tivoli Omegamon XE for IMS on z/OS

- ▶ IBM Tivoli Omegamon XE for DB2 Performance Monitor (PM)
- ▶ IBM Tivoli Omegamon for DB2 Performance Expert (PE)
- ▶ IBM Tivoli Omegamon XE for Messaging (monitor for WebSphere MQ and Message Broker)
- ▶ IBM Tivoli Omegamon XE for Storage on z/OS

The Tivoli Omegamon XE products also interface with a Tivoli Enterprise Portal Server (TEPS) which collects performance data real-time from agents on the mainframe and formats the data in predefined work spaces and views. The TEPS can also store short-term history in a data warehouse, and this data can then be used as input to performance reports or reflected in custom work spaces and views. Within TEPS, thresholds can be set for various system or subsystem conditions or resource utilizations via user-written situations. If any of these thresholds are met or surpassed, a message can be sent to the *Tivoli Enterprise Console*® (TEC) and/or an action can be taken in an automated fashion. These actions could include either issuing a console command, cutting a problem ticket, sending an email, or getting a support person paged to look at the problem immediately.

The use of thresholds and user-defined situations provides a means for being proactive in detecting potential problems before performance bottlenecks or system outages occur, thereby improving availability.

4.12.2 Capacity management and sizing

Sizing involves determining the resources required to migrate the distributed application or workloads in the high availability environment to run on the System z platform (z/OS or Linux on System z) based on current resource usage, peak transaction rate, batch elapsed times, and any service level agreements (SLAs) in place.

Capacity management involves projecting resource requirements to absorb future growth demands of these workloads on System z in a timely and cost effective manner. In addition, an effective capacity management process should utilize capacity planning tools and provide a formal way to gather, analyze and model the relevant data in order to enhance and optimize availability. Effective capacity management should ensure that:

- ▶ Response times and batch turnaround are maintained.
- ▶ SLAs are met.
- ▶ Performance problems are kept to a minimum.
- ▶ Capacity related business service outages are eliminated.

Sizing and capacity planning are covered in more detail in Chapter 8, "System z capacity planning, sizing, and TCO" on page 187.

4.12.3 Security management

Security management involves logon and password validation to online systems as well as protection against unauthorized access to facilities, files and databases that could impact availability. It is mentioned here for completeness under systems management, but this topic is really explored in detail under "Security" on page 85.

4.12.4 Service level management

Service level management involves implementing *Service Level Objectives (SLOs)* to meet the needs of the business. It could involve setting a response time goal for a given workload, an end time for completion of a batch job stream, uptime for an online application, or availability of a resource, such as the System z processor itself. Other system management disciplines would be involved in maintaining these SLAs such as capacity planning, performance management, and availability management.

An effective service level management solution would define required SLAs and generate reports. It would also track SLA data, generate warnings based on trending when SLA violations are likely to occur (to allow for proactive resolution of potential problems) and, of course, identify when SLAs are not met.

There are several service level management packages available. The *IBM Tivoli Service Level Advisor (TSLA)* is one example of a service level management solution that provides all the capabilities outlined above and more.

4.12.5 Problem and change management

Problem and change management are two distinct disciplines, but they are often treated together. Problem management is the process of managing problems from detection to final resolution. Change management, on the other hand, involves managing and controlling the introduction of change into the System z system environment.

Problem management

Problem management encompasses the detection, analysis, recovery, resolution and tracking of problems that may originate in hardware, software (operating system and application code), microcode, media (DASD, tape, and so on) or communications (networking), environmentals, or even user procedures. It includes the establishing of policies, creation of the process for dealing with problems and implementation and support of problem management software to aid in the documentation and tracking of problems. It also encompasses use of root cause analysis, which may be required depending on the severity of a problem.

The objective of problem management is to reduce the number and duration of outages and improve availability.

Key aspects of managing a problem include:

- ▶ Opening a problem ticket
- ▶ Determining resources that were affected and the impact to the business
- ▶ Establishing the proper severity
- ▶ Ensuring that proper support personnel are working on the problem and ownership has been assigned
- ▶ Determining the cause and whether the problem could have been prevented
- ▶ Restoring service as quickly as possible
- ▶ Resolving the problem within the prescribed timeframe, depending on severity
- ▶ Documenting steps taken for resolution
- ▶ Identifying whether it is a duplicate problem
- ▶ Closing the problem ticket

From a control and reporting perspective, the following functions are often performed as part of the problem management process:

- ▶ Measure the cost of any outage based on service level agreements or service contracts
- ▶ Perform trend analysis on types of problems
- ▶ Ensure that root cause analysis is performed when required
- ▶ Track problem aging
- ▶ Escalate unresolved problems
- ▶ Generate management reports
- ▶ Determine the effectiveness of the problem management process and recommend and implement changes as required

Change management

Change management encompasses planning, scheduling, coordinating, installing and monitoring changes to hardware, software (operating system and application code), microcode, media (DASD, tape, and so on) or communications (networking), environmentals, or even user procedures. It includes the establishing of policies, creation of the process for dealing with change and implementation and support of change management software to aid in the documentation and tracking of changes.

The objective of change management is to introduce change without disrupting services or to minimize the impact of change where disruption is unavoidable, thereby improving and maximizing availability.

Key aspects of managing a change include:

- ▶ Opening a change ticket
- ▶ Determining the type of change (major, minor, business as usual, and so on) based on impact to the business, infrastructure, or environment
- ▶ Planning and coordinating the change based on change type so that major changes are documented and ticket opened with more advanced notice than minor changes
- ▶ Documenting the steps involved with making the change
- ▶ Identifying planned outage duration in terms of dates and times
- ▶ Documenting the likely outcome of the change
- ▶ Identifying all areas involved in the change
- ▶ Documenting backout plans
- ▶ Ensuring that the change is completed in requested or assigned change window
- ▶ Documenting the status of the change as successful or failed
- ▶ Closing the change ticket within the prescribed timeframe after completion of the change

From a control and reporting perspective, the following functions are often performed as part of managing the change management process:

- ▶ Coordination of changes to eliminate conflicts
- ▶ Formal approval of change tickets
- ▶ Ensure that change tickets have been classified in terms of type based on risk assessment (major, minor, and so on) correctly and contain proper documentation, including backout and outage information
- ▶ Generate management reports

- Determine the effectiveness of the change management process and recommend and implement changes as required

There are many problem and change management tools available that provide the capabilities outlined above, including the *IBM Tivoli Service Request Manager*®.

4.12.6 Configuration management

The *configuration management* discipline involves identifying, recording, evaluating, tracking, coordinating, reporting, and controlling of *Configuration Items (CI)* by performing activities that maintain the integrity of these items, including their versions, constituent components and relationships throughout the life cycle of a project or on an ongoing basis. A CI is any item that can be individually managed and versioned and has been placed under configuration management control. CIs can include (but are not limited to) hardware, software, documentation, and the physical relationships and logical dependencies between these CIs.

The goals of configuration management are to ensure the integrity of a configuration item, make its evolution more manageable during updating by utilizing versions, and improve availability.

It is important that the performing organization or project has a clearly defined configuration management process in place to manage the unique complexities of each product for configurations being implemented. This allows for the ability to build, deploy, correct, and update items and, if necessary, recreate earlier versions.

One area where configuration management could be used in a System z environment is to manage the connectivity of I/O devices among CPCs and LPARs in a single or dual site. See the operational model hardware configuration examples in Chapter 5, “Operational models for high availability” on page 129. Configuration management could also be used to manage the installation and updating of system software.

There are many tools available to help implement a configuration management process. The *IBM Tivoli Configuration Manager* and *System Modification Program Extended (SMP/E)* are two examples. SMP/E is a common installation tool used for managing System z operating system components as well as middleware. It can manage multiple versions, allowing for updates and reverting to the previous version in case of problems.

4.12.7 Release management

Release management is a process that oversees the development, testing, deployment and support of software releases. It usually begins in the development cycle with requests for changes or new features. Once the request is approved, the new release is planned and designed. The release is then built, reviewed, tested and tweaked until it is ultimately accepted as a release candidate. Subsequently, the release is implemented and then bug reports and other issues are collected, generating new requests for changes. This starts the cycle all over again.

Other aspects of release management include:

- Carrying out back-out plans to remove the new version if necessary
- Informing and training clients or users about the functions of the new release

There are products and tools available to help implement and maintain a release management process. Some tools provide support for release management as well as some of the other systems management functions already discussed earlier. For example, the Tivoli

Service Request Manager provides support for the following systems management disciplines:

- ▶ Service level management
- ▶ Problem and change management
- ▶ Configuration management
- ▶ Release management
- ▶ Incident management
- ▶ Asset management

4.12.8 Storage management

Storage management is the process of managing external storage resources such as tape, DASD or virtual tape (VTS) subsystems. It involves optimizing use of resources, providing data redundancy for high availability and resolving storage related issues. For simplicity, the discussion in this section is limited to DASD resources. A comprehensive storage management solution would contain policies and procedures for other storage resources such as tape and virtual tape.

Some key aspects of establishing and maintaining an effective storage management solution include allocation and management of data on unused capacity and the optimizing of existing capacity by establishing a tiered level of resources. Under z/OS, allocation and management of new files can be controlled automatically at the system level by a *system managed storage (SMS)* policy. SMS utilizes storage pools and classes for managing data. In particular, SMS uses user-written automatic class selection (ACS) routines to assign files to the correct data class, storage class, management class, and storage group. These classes determine file placement and are also used to manage frequency of backup as well as retention and archiving to another tier, depending on frequency of access.

A tiered storage resource solution, for example, could contain three tiers, where tier 0 would be defined as regular DASD, tier 1 would be a DASD pool where data was archived in a compressed format, and tier 2 would contain files archived from tier 2 DASD to tape. Seldom-used files could migrate from tier to tier and end up on tape (tier 3) based on established access criteria defined in an SLO. However, at any time, if access to these archived files is required, they could get recalled and migrated back to tier 1 for use by a requesting job or workload. The establishment of a tiered system of storage resources is key in optimizing existing capacity and controlling storage resource costs.

There are also workloads that dictate that SMS not be used for allocation and management of data. z/OS system-related volumes often fall into this category. Non-SMS-managed volumes would be managed by separate processes outside of SMS, but in many cases can be automated with jobs scheduled by scheduling packages such as TWS.

Redundancy of data is another key aspect of storage management. Providing redundancy is important in maintaining high availability. This topic was discussed previously. See storage resilience considerations in "Storage resilience considerations" on page 80.

Storage management can also involve miscellaneous storage issues such as:

- ▶ Special capacity requirements
- ▶ Allocation and initialization of volumes with special management requirements
- ▶ Space management routines that consolidate multiple extents at the file (data set) level and unused fragmented space at volume level

- ▶ Maintenance of SMS pools
- ▶ Data recovery
- ▶ Cleanup of temporary storage resources (work packs)
- ▶ Disaster recovery planning
- ▶ Performance of storage devices which would involve performance management to optimize use and availability, and possibly the use of performance tools such as IBM Tivoli Omegamon XE for Storage on z/OS
- ▶ Tracking storage capacity versus utilization

There are tools that can help automate storage management functions. One common tool is the *IBM Data Facility Hierarchical Storage Manager (DFSMSHsm)*.

4.12.9 Service continuity management

The purpose of service continuity management is to address the IT services that will be required to support recovery activities according to a committed recovery schedule or SLA if a disruption to business occurs. Planning to ensure business continuity plays a key role in a high availability environment. Service continuity management includes anticipating incidents that might cause disruption to critical business functions and the planning on how to handle such incidents. This would include identifying the IT areas involved in handling an unexpected outage and development of strategies and plans for dealing with the outage so that committed recovery time frames can be met.

In a high availability environment, a fast or immediate recovery strategy would be required. This can best be implemented with an automated solution such as what GDPS would provide in a Parallel Sysplex environment or GDOC for a Linux on System z environment. See "Availability and scalability" on page 75 dealing with disaster recovery.

4.12.10 Incident management

Incident management involves the management of problems from an incident perspective. An incident can be defined as any interruption or reduction in quality of service. Incident management is closely related to problem management discussed earlier.

Incident management deals with restoring normal service operation as quickly as possible while minimizing the impact on business operations, thereby maximizing availability. Problem management, on the other hand, deals with resolving the root cause of incidents with either a workaround or permanent solution.

Incident management can be implemented at the service desk level and/or with special tools or packages (such as the Tivoli Service Request Manager) and typically involves multiple steps in dealing with any particular incident. Regardless of the method used for incident management, it is essential that each step for an incident be well documented.

Here are typical steps that can be followed in dealing with an incident:

- ▶ Identification and logging of the incident.
- ▶ Classification of an incident into proper category assigning proper priority based on urgency and impact to the business. Some packages may have rules established that autonomically assign the incident to a particular service group for resolution based on the incident classification.
- ▶ Initial diagnosis of the incident.

- ▶ Escalation of the incident over time to ensure that it is being handled with proper attention. Escalation can be based on predefined rules in terms of an existing SLA for incident category or priority or inactivity (elapsed time without an update being logged).
- ▶ Investigation and diagnosis.
- ▶ Resolution and recovery.
- ▶ Closure.

In any incident management process, there should be periodic reviews of the incident history to promote continual improvement to the process. These improvements could take the form of refinements to rules for assigning incidents or modifications to incident SLAs or inactivity thresholds.

4.12.11 Asset and financial management

The purpose of asset and financial management is to ensure that financial control and procedures are in place to effectively forecast and control budgets, enable business decisions and ensure that proper business compliance (legal, corporate, regulatory) is maintained.

Asset management as a whole typically addresses hardware and software assets throughout their entire life cycle from acquisition to retirement. It involves identifying, tracking and managing software licenses as well as hardware inventory and contracts.

An effective software asset solution is important to help control unauthorized purchases and mitigate legal or financial risk for non-compliant software that may be installed. An effective hardware asset solution entails management of physical components of computer equipment and networks.

Financial management, on the other hand, involves gathering of financial information that can aid an organization in making business decisions related to these hardware and software assets based on financial objectives. It can also involve budgeting, reporting on resource usage, and IT accounting of relative amounts spent on IT services, as well as chargeback and billing. In addition, Financial management can help provide information about where cost savings can be gained.

There are many tools that can help implement asset and financial management processes. For an asset management solution, examples are *IBM License Compliance Manager* or *Tivoli Service Request Manager*. For a financial management solution, an example is *IBM Tivoli Usage and Accounting Manager (TUAM)*.

4.13 Specialty processors

An IBM System z mainframe is often called a *Central Processor Complex (CPC)* to refer to the physical collection of hardware that includes main storage, one or more central processors, timers, and channels. Further, the processors in the CPC are called *Processing Units (PUs)*. When IBM delivers a CPC, the PUs are characterized as follows:

- ▶ General CPs (for normal work)
- ▶ System Assist Processor (SAP)
- ▶ Integrated Facility for Linux (IFL)
- ▶ Integrated Coupling Facility (ICF) for Parallel Sysplex configurations
- ▶ System z Application Assist Processor (zAAP)

► System z Integrated Information Processor (zIIP)

Important: All the above PUs except General CPs are considered specialty processors, since they are targeted to process only certain kinds of work.

The intent of specialty engines (with the exception of SAP) is to add capacity without increasing software costs and to offload specialized work to take advantage of these special PUs. This frees up capacity for growth of workloads on the general CP. In addition, the cost of adding a specialty processor to a CPC is usually less expensive than adding capacity with a general CP.

Figure 4-21 shows the evolution over time.

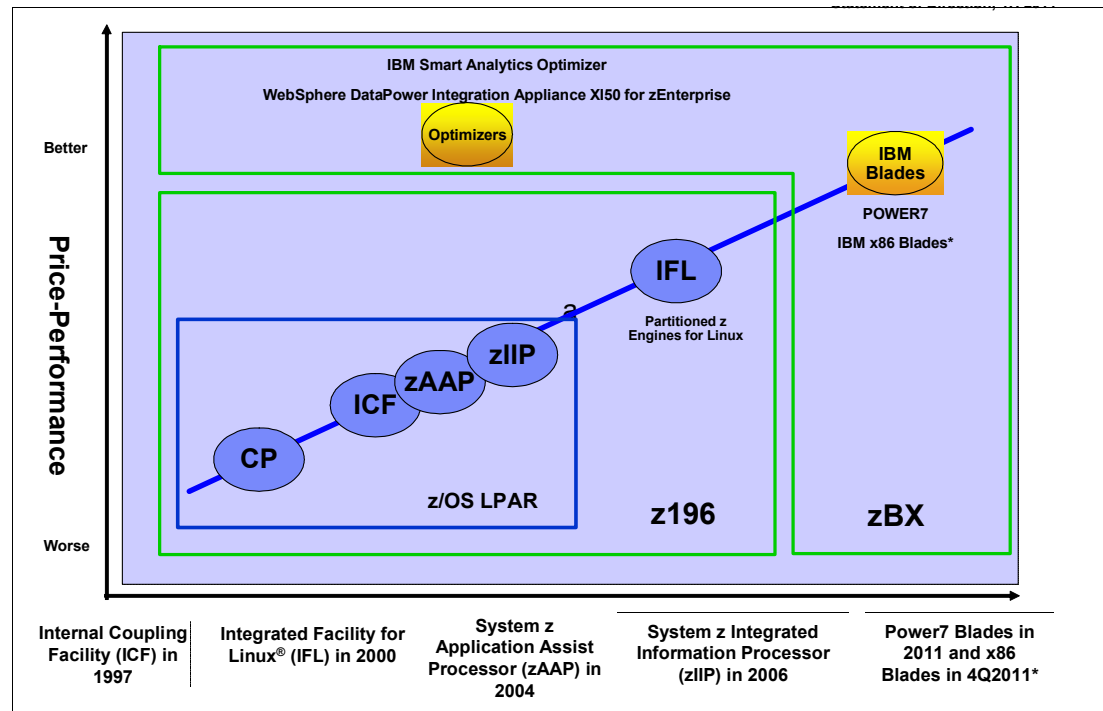


Figure 4-21 The evolution of specialty processors

System Assist Processor (SAP)

SAP is a dedicated I/O processor that comes standard on every CPC. It is used to help improve efficiency and reduce the overhead of I/O processing for all the logical partitions on the CPC, regardless of the operating system (z/OS, z/VM, or zLinux). For very high I/O-intensive workloads, additional SAPs can be purchased (model dependent).

Integrated Facility for Linux (IFL)

IFLs are used to support a Linux workload on a System z server under z/VM. With the new z196 CPC, the cost of an IFL has been significantly reduced, making Linux significantly less expensive to run versus the z9 and z10 processors.

The Integrated Facility for Linux (IFL) is a central processor (CP) that is dedicated to Linux workloads. IFLs are managed by PR/SM in logical partitions with dedicated or shared processors. The implementation of an IFL requires a logical partition (LPAR) definition, following the normal LPAR activation procedure. An LPAR defined with an IFL cannot be

shared with a general purpose processor. IFLs are supported by z/VM, the Linux operating system and Linux applications, and cannot run other IBM operating systems.

A Linux workload on the IFL does not result in any increased IBM software charges for the traditional System z operating system and middleware.

Figure 4-22 shows that adding IFL capacity to your mainframe does not increase the IBM software licensing costs for the traditional mainframe environment (for example, z/OS).

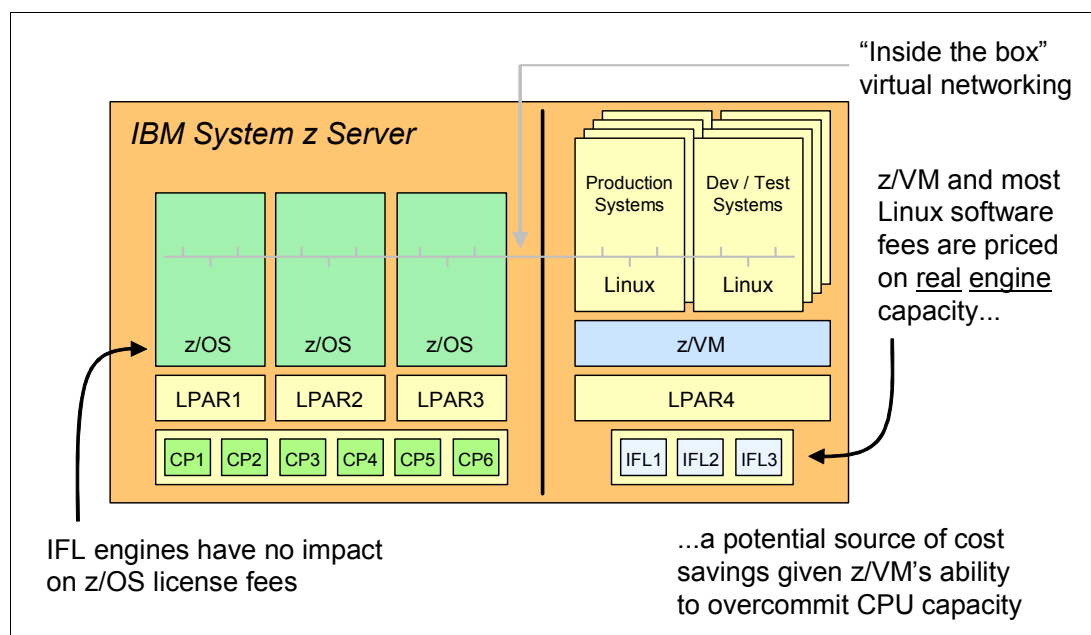


Figure 4-22 Adding IFL capacity to mainframe does not increase the IBM software licensing costs

Internal Coupling Facility (ICF)

ICF enables multiple z/OS LPARs to share, cache, update, and balance data access. This hardware is a major component of Parallel Sysplex. ICFs also allow Internal Coupling (IC) links to help eliminate the requirements for external CF links.

System z Application Assist Processor (zAAP)

zAAP reduces the standard processor (CP) capacity requirements for z/OS Java or XML System Services applications, freeing up capacity for other workload requirements.

Apart from the cost savings, the integration of new applications with their associated database systems and transaction middleware (such as DB2, IMS, or CICS) can simplify the infrastructure, for example, by introducing a uniform security environment, reducing the number of TCP/IP programming stacks and server interconnect links. Furthermore, processing latencies that would occur if Java application servers and their database servers were deployed on separate server platforms are prevented.

System z Integrated information Processor (zIIP)

The zIIP specialty processor is designed to help free up general computing capacity and lower overall total cost of computing for the specific data and transaction processing workloads for business intelligence (BI), ERP, and CRM), and the specific network encryption (zIIP Assisted IPsec) workloads on the mainframe. The current direction is for more workloads to become eligible to exploit use of zIIP engines.

Here is a list of workloads currently supported by zIIPs:

Four types of DB2 work:

- Remote DRDA® access via TCP/IP including access across LPARs via HiperSockets.
 - Parallel query operations. DB2 V9 can increase the amount of parallel processing and thus use the zIIP more.
 - XML parsing in DB2 (can use zIIP or zAAP processors).
 - Certain types of DB2 utilities.
- ▶ IPSEC in TCP/IP.
 - ▶ HiperSockets for large messages. The z/OS Communications Server allows the HiperSockets Multiple Write Facility processing for outbound large messages originating from z/OS to be performed on a zIIP.
 - ▶ Certain general XML processing.
 - ▶ z/OS Global Mirror (zGM, formerly XRC Extended Remote Copy) System Data Mover processing associated with zGM/XRC.
 - ▶ IBM Scalable Architecture for Financial Reporting (SAFR).
 - ▶ Some Common Information Model (CIM) processing. CIM is used by z/OS to communicate information about or manage resources for its system components. As of z/OS 1.11, CIM client applications such as parts of the System z Capacity Provisioning Manager and parts of the z/OS Management Facility can benefit from zIIP.
 - ▶ Certain third party vendor software products.

zAAP on zIIP

z/OS R11 added a new capability (*zAAP on zIIP*) allowing zIIP- and zAAP-eligible workloads to run on the zIIP engine. This new capability is ideal for environments without enough zAAP- or zIIP-eligible workloads to justify a specialty engine today but the combined eligible workloads may make the acquisition of a zIIP more cost effective. This new capability could also provide more value for environments having only zIIP processors by making Java and XML-based workloads eligible to run on existing zIIPs. zAAP on zIIP capability is not available for z/OS LPARS if zAAPs are already installed on the CPC.



Operational models for high availability

As we have discussed earlier, there can be a significant business impact associated with an IT outage. The potential loss of revenue associated with a one-hour IT outage is estimated to be from 1 to 3 million dollars for the energy, manufacturing, financial services and retail industry sectors. The business impact is even greater considering the consequences of lower customer satisfaction and a tarnished reputation in the market. For example, what about incidents involving slow website response times that may not be classified as an outage but more of an IT capacity constraint?

The root cause of the IT outage could be anything from human error, unplanned hardware outage or natural disaster. It is not a question of *if* these events will occur but *when* they will occur. When these events do occur in the IT environment, how will the impact be masked or minimized such that the business service remains available?

In this chapter we highlight some of the operational models that provide the high availability, continuous operations and disaster recovery attributes that reduce the odds of having an outage. These operational models represent the System z specific implementations of the target reference architectures identified in Chapter 3, “Target architectures for availability” on page 15.

5.1 Introduction

In Chapter 2, “Introduction to systems availability” on page 3 we defined and discussed high availability, continuous operations and continuous availability in detail. Disaster recovery is the process, policies and procedures related to preparing for recovery or continuation of technology infrastructure critical to an organization after a natural or human-induced disaster.

How do we achieve the required levels of availability during planned and unplanned outages in the System z environment? What System z features are used to address disaster recovery scenarios such as a site outage due to a tornado, flood, widespread power outage or terrorist act? What are the options available for the z/OS and Linux operating systems?

In the models discussed in this chapter we describe the environments that address both high availability and disaster recovery. This is not intended to be an exhaustive list, but these models should be able to address most high availability and disaster recovery requirements.

5.1.1 Planned outages

Sometimes system outages are required to maintain and upgrade hardware or software. These outages can be planned to minimize impact and continue to provide high availability and continuous operation. The type of outages—LPAR, sysplex, processor or site—that can be considered in your environment will depend on the architecture of your configuration. The objective of the operational models is to mask or minimize the server, system, or application outage in such a way that the probability of a service outage is greatly reduced or eliminated.

5.1.2 A word on scalability and capacity

In a large enterprise, the scalability and capacity attributes of an IT system are key. As we highlighted in Chapter 4, “System z technology options” on page 49, System z has the ability to scale horizontally by adding LPARs and vertically by adding processor, memory, and I/O resources to an existing System z server.

Unexpected events, such as unpredictable increases in workload, or even disasters, can cause the need for IT capacity to grow tremendously. System z provides just-in-time deployment of additional computing capacity, known as *Capacity on Demand (CoD)*. The functions are designed to provide more flexibility and to provide an easier way to dynamically change capacity when business requirements dictate. Capacity on Demand is the key feature that enables you to adjust your processing and memory capacity to your specific needs without having to shut down or restart the server, and without needing to re-IPL the operating system. Additional capacity resources can be dynamically activated, either partially or in total, by using granular activation controls directly from the management console of the System z server, without having to interact with IBM Support. CoD delivers permanent or temporary capacity upgrades.

Temporary capacity upgrades

Temporary upgrades can be done by *Capacity Backup (CBU)*, *Capacity for Planned Events (CPE)*, or *On/Off Capacity on Demand (On/Off CoD)*, all by using the CIU facility on IBM Resource Link.

- Capacity Backup (CBU) is intended to replace capacity lost in the enterprise as a result of a disaster. CBU cannot be used for peak workload management. A CBU can last up to 90 days when a disaster situation occurs.

- Capacity for Planned Events (CPE) is used to replace temporary lost capacity in an enterprise for planned downtime events, for example for data center changes. CPE is not meant to be used for peak workload management, or for a disaster situation. The CPE provides for one 72-hour activation.
- On/Off Capacity on Demand (On/Off CoD) enables concurrent and temporary additional capacity on the server. On/Off CoD can be used for customer peak workload requirements. It has a daily hardware charge. The software charges vary according to the license agreement for the individual products.

5.2 Single-site Parallel Sysplex

The first operational model to discuss is shown in Figure 5-1.

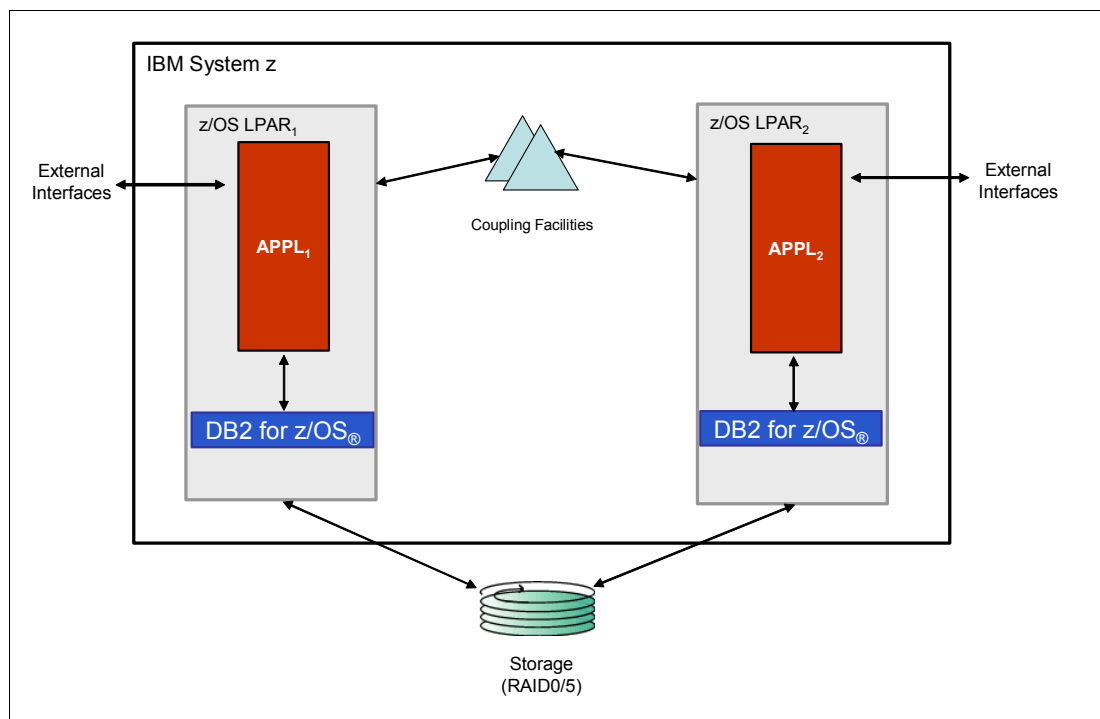


Figure 5-1 Single-site Parallel Sysplex

5.2.1 Model description

This model consists of one System z server with two z/OS Logical Partitions (LPARs) in a Parallel Sysplex. There are two Coupling Facilities which can also reside on the same System z server. The physical resources are shared by the LPARs. The application represented is a stand-alone application that is using DB2 as the database product. The application has been cloned with each LPAR running its own application image. DB2 is configured for data sharing with both instances accessing the same database. There is only one disk subsystem. Incoming network traffic is via the IP network and can be balanced using Sysplex Distributor. This model is an example of *virtual clustering* and will be expanded in subsequent models to address higher levels of availability.

5.2.2 Availability characteristics

z/OS Parallel Sysplex is a key technology for providing high availability in single-site System z environments. The model application is Parallel Sysplex enabled and can be cloned across the sysplex providing the redundancy needed to achieve the highest levels of availability. DB2 data sharing exploits the Parallel Sysplex capability and extends the redundancy to the DB2 database. The characteristics are:

- ▶ No planned service outages necessary with Parallel Sysplex, except for some HW upgrades.
- ▶ If all LPARs are on one server, then some HW upgrades require downtime.
Option: use two System z servers for even higher availability and complete freedom from planned service outage
- ▶ Multiple application images protect from single application point of failure.
- ▶ Coupling Facilities are duplexed for additional availability.

5.2.3 Functionality

- ▶ Dual LPAR in a single site.
- ▶ Single Disk Storage Subsystem is single point of failure.
- ▶ Work request balanced across both LPARs under control of WLM.
- ▶ DB2 data sharing across systems with lock, list and cache structures in the Coupling Facilities (CFs).
- ▶ Sysplex Distributor is the internal decision maker for application workload balancing (where the workload is a TCP/IP connection). A device on the IP network will see an application cluster as one Dynamic Virtual IP Address (DVIPA), regardless of the number of LPARs in the sysplex.
- ▶ The Coupling Facility is used for list, lock, and cache structures.

5.2.4 Availability scenario

How do we achieve scalability and disaster recovery in this model?

If the single site were configured with one System z server, an outage of an LPAR would result in continuous operation and high availability while an outage at the Parallel Sysplex, machine, or site level would result in a disruption of service. Failover to a redundant LPAR would occur on the same server.

If the site were configured with two or more System z servers, an outage on an LPAR or server basis would result in continuous operation and high availability while an outage at the Parallel Sysplex or site level would result in a disruption of service. Failover to a redundant LPAR would occur on another System z server in the sysplex.

Here is the sequence of events for an unplanned application outage in the model depicted in Figure 5-1 on page 131:

1. Prior to the outage, both instances of the application receive incoming requests with balancing of connections being handled by Sysplex Distributor in either a round robin or WLM-managed fashion.
2. The application instance on LPAR1 experiences a problem and ends abnormally.

3. Sysplex Distributor identifies the application instance on LPAR1 as unavailable and directs all incoming network traffic to the remaining application instance on LPAR2.
4. Since we are in a DB2 data sharing environment, the application instance on LPAR2 has access to the same DB2 tables.
5. Assuming the two z/OS LPARs were sharing 10 processors, the remaining application instance on LPAR2 would still have all 10 processors available.
6. Once the application instance on LPAR1 is restarted (either with an automation product or feature or manually), the application instance would be recognized by Sysplex Distributor and identified as available for incoming connection requests along with the application instance on LPAR2.

5.3 Single-site Parallel Sysplex with two servers and disk-level replication (basic Hyperswap)

The second operational model to discuss is shown in Figure 5-2.

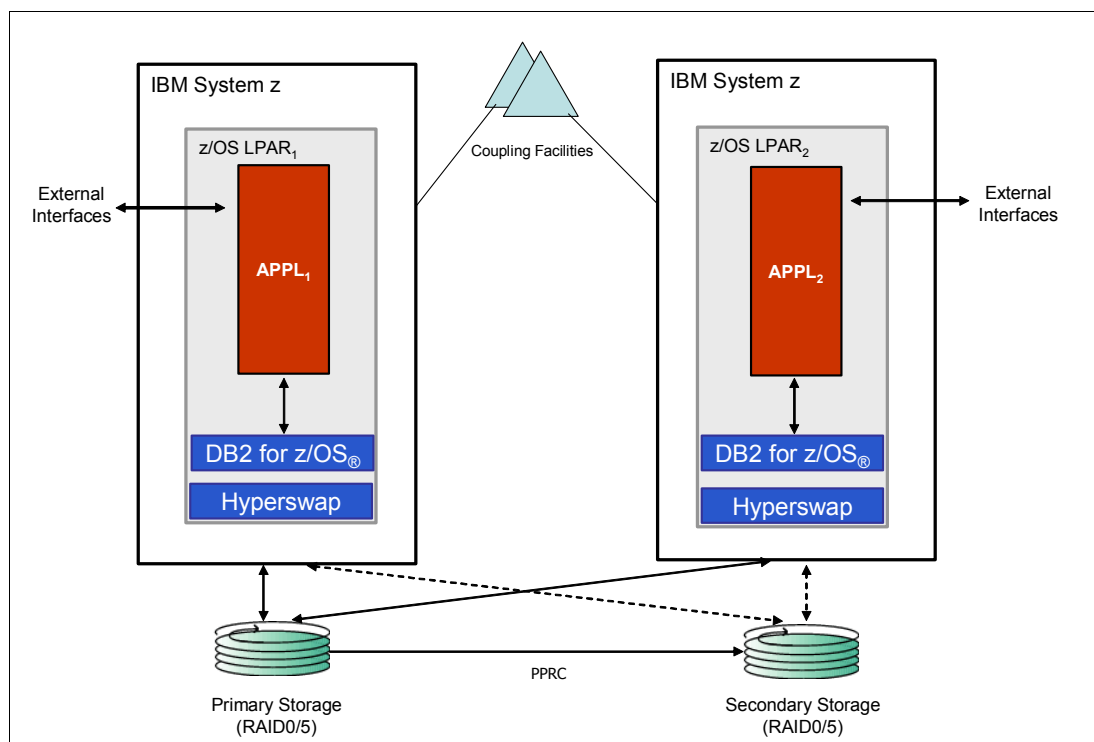


Figure 5-2 Single-site Parallel Sysplex with two servers and disk-level replication

5.3.1 Model description

This model consists of two System z servers in a Parallel Sysplex. Each System z server is running one z/OS LPAR with cloned application images. The application represented is a stand-alone application that is using DB2 as the database product. There are two disk subsystems. This model employs disk replication with Metro Mirror with Peer-to-Peer Remote Copy (PPRC). Basic Hyperswap is used to transparently switch the PPRC primary with the PPRC secondary disk subsystems.

5.3.2 Availability characteristics

The availability characteristics are:

- ▶ Two System z servers provide highest availability during normal operations and complete freedom from planned service outage (three System z servers would provide single point of failure elimination even during one server's planned outage).
- ▶ Multiple application images protect from single application point of failure.
- ▶ Redundant disks with Metro Mirror and Hyperswap broaden the continuous availability attributes of Parallel Sysplex.

5.3.3 Functionality

- ▶ DB2 data sharing
- ▶ Two Disk Storage Subsystems with Metro Mirror (Peer to Peer Remote Copy) and Hyperswap, which is managed by *TotalStorage Productivity Center for Replication (TPC-R)*.

5.3.4 Availability scenario

How do we achieve scalability and disaster recovery with this model?

If the site were configured with two or more System z servers, an outage on an LPAR or server basis would result in continuous operation and high availability while an outage at the sysplex or site level would result in a disruption of service. Failover to a redundant LPAR would occur on another System z server in the sysplex,

Here is the sequence of events for an unplanned primary disk outage:

1. Prior to the outage, both instances of the application receive incoming requests with balancing of connections being handled by Sysplex Distributor in either a round robin or WLM-managed fashion.
2. The primary disk subsystem experiences an outage.
3. Hyperswap invokes a PPRC Data Freeze, which causes all I/O to all storage devices to be queued, thus maintaining full data integrity and cross-volumes data consistency.
4. z/OS completes the HyperSwap® operation and rebuilds all z/OS control blocks to point to the secondary disk storage devices.
5. All I/O is released and the application continues to run against the secondary disk storage.

5.4 Dual site with data replication

In this operational model we introduce a second site, shown in Figure 5-5 on page 139.

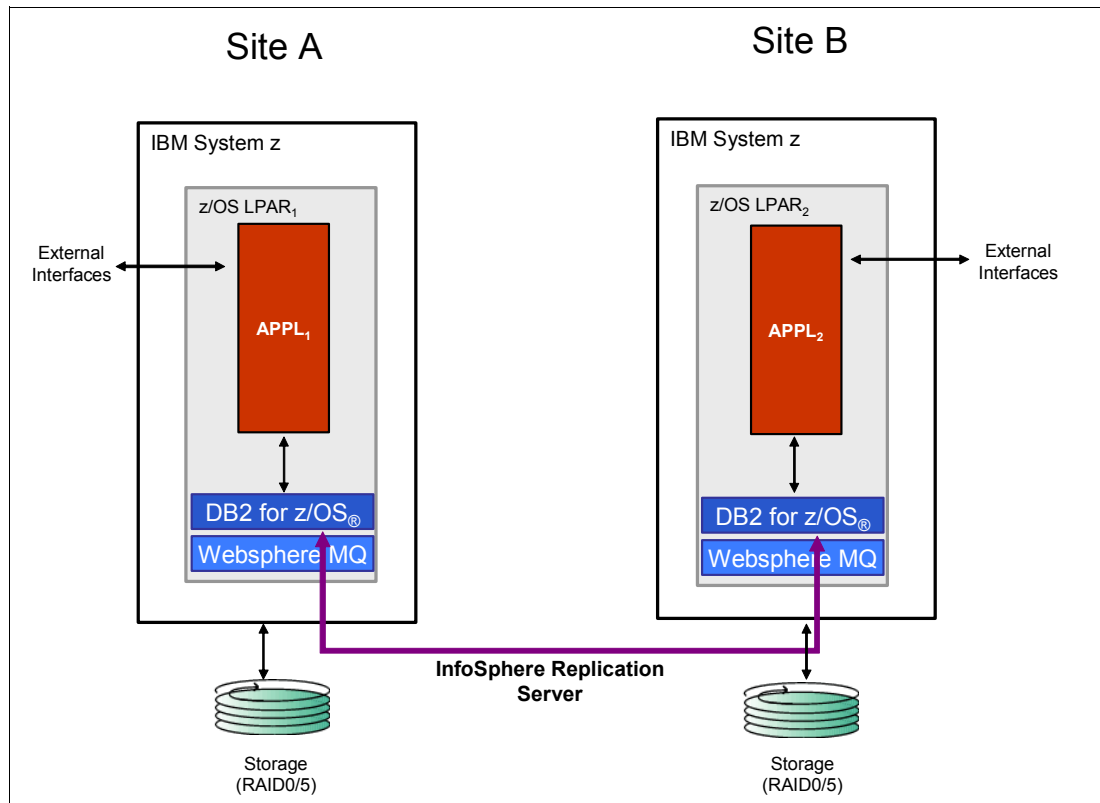


Figure 5-3 Dual site with data replication

5.4.1 Model description

This model consists of two stand-alone System z servers located at two sites with no distance restrictions. Each System z server is running one z/OS LPAR with cloned application images. The application represented is a stand-alone application that is using DB2 as the database product. DB2 is active on both sites, but the application may either be in an active or passive state on Site B. The DB2 data is replicated using InfoSphere Data Replicator, which uses WebSphere MQ to send database updates to a target system. Replication can be either unidirectional for an active/passive configuration, or bidirectional for an active/active configuration. There is only one disk subsystem per site. This model could be used for disaster recovery.

5.4.2 Availability characteristics

The availability characteristics are:

- ▶ Two System z servers provide highest availability during normal operations and complete freedom from planned service outages. (Three z servers provide SPOF elimination even during one server's planned outage.)
- ▶ Multiple application images protect from single application point of failure.
- ▶ The DR can operate either in active/passive or active/active mode.
- ▶ In active/passive mode, the secondary site is ready to take over by activating device and network communications to it upon failure of the primary site.
- ▶ Unlimited distance between sites.

5.4.3 Functionality

- ▶ Single LPAR in each site.
- ▶ Single Disk Storage Subsystem is single point of failure.
- ▶ Minimum replication latency; changes are sent to the replication server as soon as they are read from the source log.
- ▶ Multi-threaded queue replication program ensures rapid duplication of data.
- ▶ Replication messages are compressed to reduce network latency.
- ▶ Replication operations are asynchronous.
- ▶ Since the queues are persistent, replication messages can survive a system or device failure.

5.4.4 Availability scenario

An outage on an LPAR, machine, or site basis would result in continuous operation and high availability. Failover to redundant LPARs would occur on another System z machine in the second location. Databases at the respective sites are kept in sync by using InfoSphere Data Replicator. For the active/passive configuration, the application image at Site B would need to be started. InfoSphere Data Replicator would be unidirectional with the DB2 log at Site A being used to capture updates and then applying the updates to the DB2 database at Site B. For the active/active configuration, the Site B application would continue processing as usual. InfoSphere Data Replicator would be bidirectional with the DB2 updates being captured at both sites and the corresponding updates applied to the partner site DB2 database.

Here is the sequence of events for an unplanned System z outage at Site A. In this example, we use an active/active scenario:

1. Both instances of the application receive incoming requests with balancing of connections being handled at the network router level. The InfoSphere Data Replicator Capture program reads the DB2 recovery log for changes to a source table that is to be replicated. The program then sends transactions as messages over queues, where they are read and applied to target tables by the Apply program. In an active/active scenario, any DB2 source table updates at Site A would be sent replicated on Site B and vice versa.
2. System z server LPAR1 at site A experiences a power outage.
3. The router now directs all incoming network traffic to the application instance at Site B LPAR2.
4. Since we are using InfoSphere Data Replicator, the application instance at Site B LPAR2 has access to the latest updates to the DB2 tables.
5. The Site B LPAR2 System z server may have to employ a feature such as Capacity Backup (CBU) to provide the additional resources necessary to handle the workload formerly handled by two servers.
6. Once the application instance on LPAR1 is restarted (either with an automation product or feature, or manually), InfoSphere Data Replicator resumes the capture and applying of database updates at both sites.

5.5 Dual-site Parallel Sysplex with data-level replication (active/passive)

As depicted in Figure 5-5, this operational model adds a Parallel Sysplex capability to both sites.

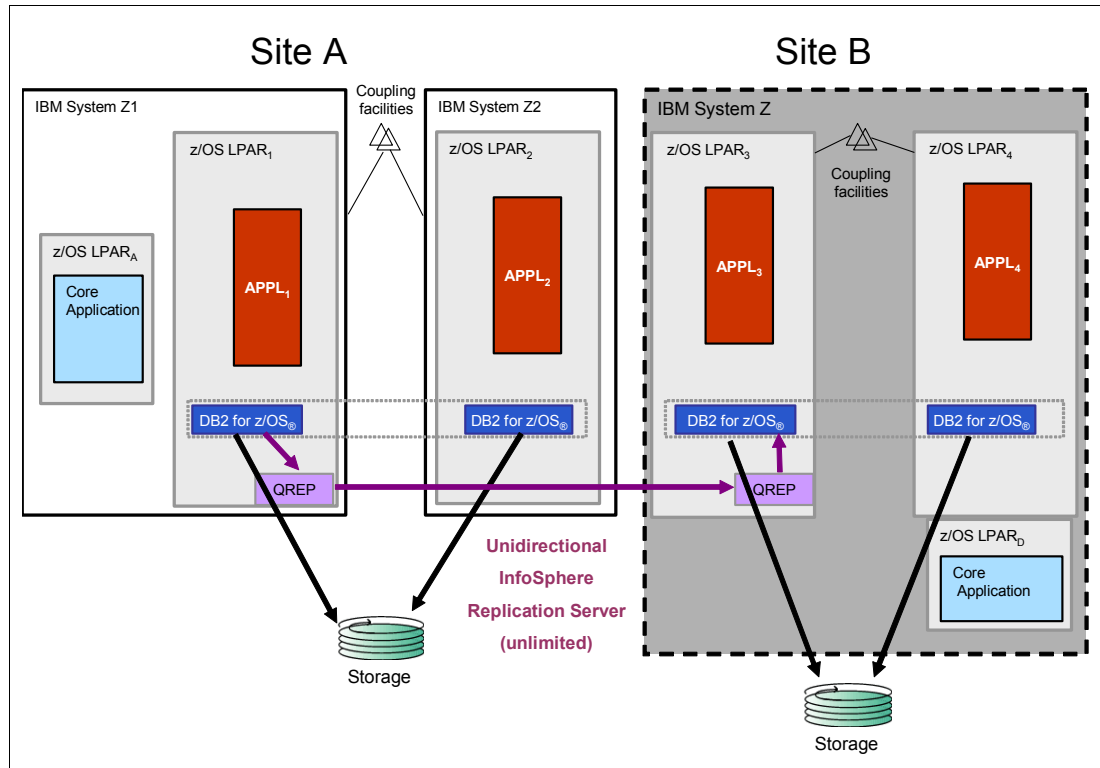


Figure 5-4 Dual-site Parallel Sysplex with data level replication

5.5.1 Model description

In this model we have two sites, each with its respective Parallel Sysplex environment. This model utilizes both physical and virtual clustering techniques. The sample application is a stand-alone application using DB2 as the database manager. DB2 is configured for data sharing with both instances within the sysplex accessing the same database. This is an active/passive configuration with Site A as the active site and Site B as the passive site. DB2 is active on both sites, but the application is in a passive state on Site B. The DB2 data is replicated using InfoSphere Data Replicator, which uses WebSphere MQ to send database updates to a target system. Replication in this model is unidirectional. All of the database updates at Site A are captured and applied at Site B. Since the databases between Site A and Site B are synchronized, the application can be in a warm state in which it is active but not receiving any inbound requests from the network.

5.5.2 Availability characteristics

The availability characteristics are as follows:

- ▶ Two System z servers provide highest availability during normal operations and complete freedom from planned service outages (three z servers provide SPOF elimination even during one server's planned outage).

- ▶ Multiple application images protect from single application point of failure.
- ▶ In active/passive mode, the secondary site is ready to take over by activating device and network communications to it upon failure of the primary site.
- ▶ Unlimited distance between sites.

5.5.3 Functionality

- ▶ Dual LPAR in each site.
- ▶ DB2 data sharing on each site.
- ▶ Single Disk Storage Subsystem is single point of failure.
- ▶ Minimum replication latency: Changes are sent to the replication server as soon as they are read from the source log.
- ▶ Multi-threaded queue replication program ensures rapid duplication of data.
- ▶ Replication messages are compressed to reduce network latency.
- ▶ Replication operations are asynchronous.
- ▶ Since the queues are persistent, replication messages can survive a system or device failure.
- ▶ Sysplex Distributor is the internal decision-maker for application workload balancing (where the workload is a TCP/IP connection). A device on the IP network will see an application cluster as one Dynamic Virtual IP Address (DVIPA), regardless of the number of LPARs in the sysplex.

5.5.4 Availability scenario

An outage on an LPAR, machine, sysplex, or site basis at Site A would result in continuous operation and high availability. Failover to redundant LPARs would occur on another System z machine in Site B. Databases at the respective sites are kept in sync by using InfoSphere Data Replicator. For the active/passive configuration, the application image at Site B would need to be started. InfoSphere Data Replicator would be unidirectional with the DB2 log at Site A being used to capture updates and then applying the updates to the DB2 database at Site B.

Here is the sequence of events for an unplanned Parallel Sysplex outage at Site A. In this example, we use an active/passive scenario:

1. Only the Site A instances of the application are receiving incoming requests with balancing of connections being handled by Sysplex Distributor. The InfoSphere Data Replicator Capture program at Site A reads the DB2 recovery log for changes to a source table that is to be replicated. The program then sends transactions as messages over queues, where they are read and applied to target tables by the Apply program at Site B.
2. Site A experiences a power outage.
3. Site B now assumes the role as primary site with the router directing all incoming network traffic to the application instances at Site B.
4. Additional LPARs and application instances at Site B are started either manually or by an automations product.
5. Since we are using InfoSphere Data Replicator, application instances at Site B have access to the latest updates to the DB2 tables.

5.6 Dual-site Parallel Sysplex with disk-level replication (Metro Mirror)

This model, as depicted in Figure 5-5, adds disk replication.

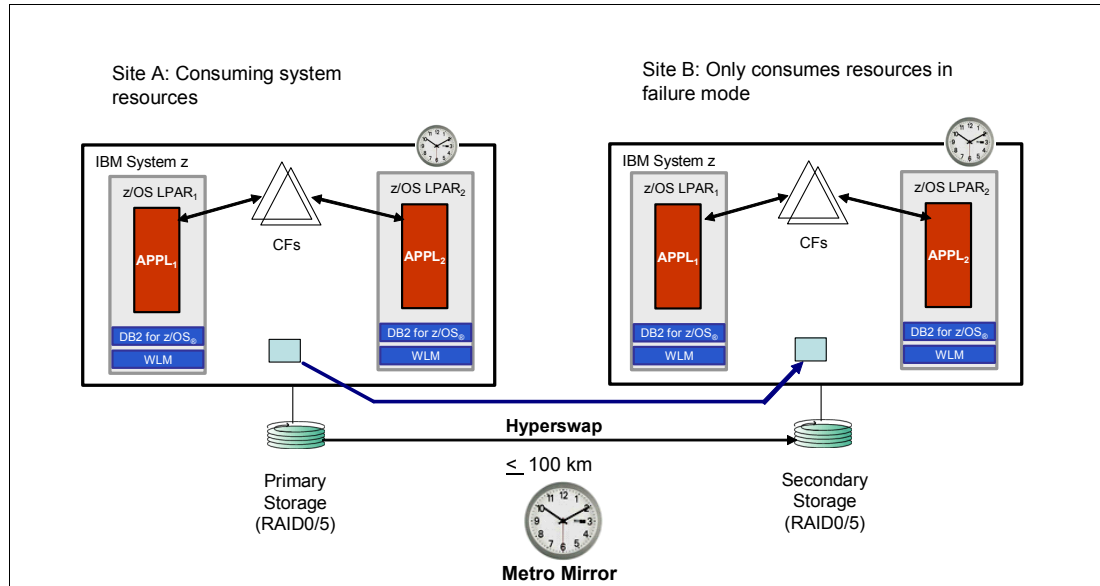


Figure 5-5 Dual-site Parallel Sysplex with disk-level replication

5.6.1 Model description

The System z servers in this model are located at two sites with a distance of less than 100 km. This is a Parallel Sysplex environment with each System z server running two z/OS LPARs with cloned application images. The application represented is a stand-alone application that is using DB2 as the database product. There are two disk subsystems. This model employs disk replication with Peer to Peer Remote Copy (PPRC). Hyperswap is used to transparently switch the PPRC primary with the PPRC secondary disk subsystems.

5.6.2 Availability characteristics

The availability characteristics are as follows:

- ▶ No planned outages are necessary with Parallel Sysplex and DB2 data sharing, except for some HW upgrades.
- ▶ Option: use two System z servers per site for even higher availability and complete freedom from planned service outages.
- ▶ Coupling Facilities are duplexed.
- ▶ Multiple application images protect from a single application point of failure.
- ▶ Disaster recovery based on active/stand-by.

5.6.3 Functionality

- ▶ Dual LPAR; dual site model (Parallel Sysplex configuration required).
- ▶ Metro Mirror provides remote data copy at the disk subsystem layer.

- ▶ Distance between sites is <100 km.
- ▶ A Site 1 failure or communications failure invokes Hyperswap and Coupling Facility takeover at Site 2.
- ▶ DB2 databases shared across systems with lock, list, and cache structures in the Coupling Facilities (CFs).
- ▶ Sysplex Distributor is the internal decision maker for application workload balancing (where the workload is a TCP/IP connection). A device on the IP network will see the application cluster as one Dynamic Virtual IP Address (DVIPA), regardless of the number of LPARs in the sysplex.
- ▶ Communication to the host application can be via synchronous, asynchronous, TCP/IP, MQ, or any other mechanism.

5.6.4 Availability scenario

An outage on an LPAR, sysplex, machine, or site basis would result in continuous operation and high availability. Failover would occur to one or more redundant LPARs at the second site, depending on the type of outage (LPAR, machine, or site).

Here is the sequence of events for an unplanned application outage:

1. Both instances of the application receive incoming requests with balancing of connections being handled by Sysplex Distributor in either a round robin or WLM-managed fashion.
2. The application on LPAR1 experiences a problem and ends abnormally.
3. Sysplex Distributor now directs all incoming network traffic to the application instance on LPAR2.
4. Since we are in a DB2 data sharing environment, the application instance on LPAR2 has access to the same DB2 tables.
5. Assuming the two z/OS LPARs were sharing 10 processors, the remaining application instance on LPAR2 would still have all 10 processors available.
6. Once the application instance on LPAR1 is restarted (either with an automation product or feature, or manually), the application instance would be recognized by Sysplex Distributor and identified as available for incoming connection requests along with the application instance on LPAR2.

5.7 Dual-site Parallel Sysplex with disk-level replication (Global Mirror)

This model, shown in Figure 5-6 on page 141, is a variation of the model discussed in “Dual-site Parallel Sysplex with disk-level replication (Metro Mirror)” on page 139, but uses another type of disk-level replication.

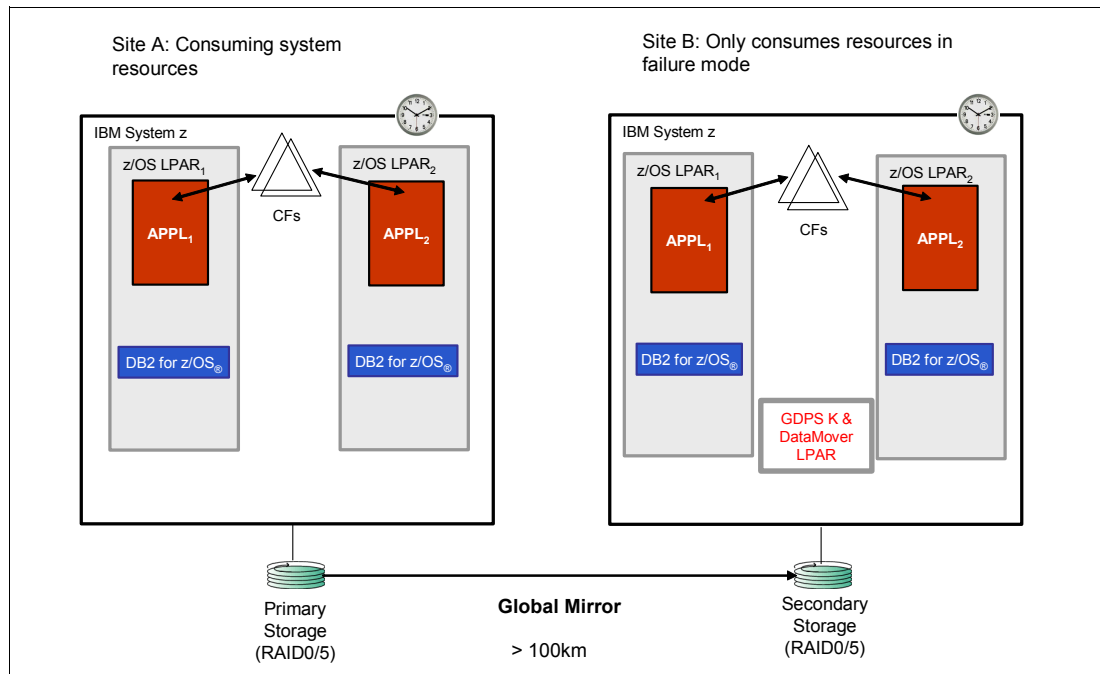


Figure 5-6 Dual-site Parallel Sysplex with disk-level replication (Global Mirror)

5.7.1 Model description

This model consists of two System z servers in a Parallel Sysplex. The System z servers in this model are located at sites separated by unlimited distance. Each System z server is running two z/OS LPARs with cloned application images. The application represented is a stand-alone application that is using DB2 as the database product. There are two disk subsystems. This model employs disk replication with Global Mirror.

5.7.2 Availability characteristics

The availability characteristics are as follows:

- ▶ All work is normally handled by the primary site.
- ▶ No planned outages necessary with Parallel Sysplex and DB2 data sharing, except for some HW upgrades.

Option: use two System z servers per site for even higher availability and complete freedom from planned service outages.

- ▶ Coupling Facilities are duplexed.
- ▶ Multiple application images protect from a single application point of failure.
- ▶ Disaster recovery based on active/stand-by.

5.7.3 Functionality

- ▶ Dual LPAR; dual-site model (Parallel Sysplex configuration required).
- ▶ Global Mirror provides remote data copy at the disk subsystem layer.
- ▶ Unlimited distance between sites.

5.7.4 Availability scenario

Here is the sequence of events for an unplanned application outage:

1. Prior to the outage, only Site A instances of the application receive incoming requests with balancing of connections being handled by Sysplex Distributor in either a round robin or WLM-managed fashion.
2. Site A experiences a site-wide power outage.
3. GDPS K-sys identifies the outage, which triggers the start of all LPARs and applications on Site B.
4. HyperSwap invokes a PPRC Data Freeze, which causes all I/O to all storage devices to be queued, thus maintaining full data integrity and cross volumes data consistency.
5. z/OS completes the HyperSwap operation and rebuilds all z/OS control blocks to point to the secondary disk storage devices.
6. All I/O is released and the application continues to run against the secondary disk storage.

5.8 Linux on System z - Linux HA

In this section we discuss the first of two models for the Linux on System z environment.

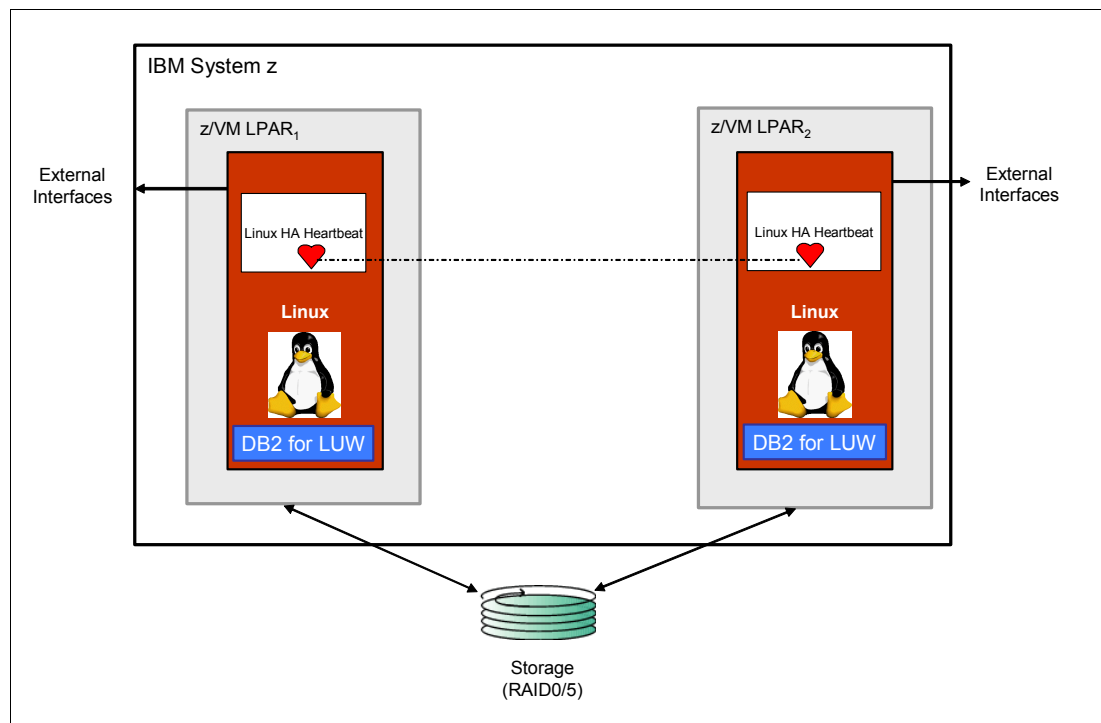


Figure 5-7 Linux on System z - Linux HA

5.8.1 Model description

This model consists of two z/VM LPARs, each hosting one or more Linux images on a single System z server. This is an active/passive model with DB2 running on the Linux guest on LPAR1. This is a shared storage configuration with the DB2 instance and database residing in the shared file system. There is only one disk subsystem. This model uses the Linux-HA

support to provide high availability. The *Linux-HA* project provides HA solutions for Linux through an open development community. The core of Linux-HA Release 2 is a component called *Heartbeat*. Heartbeat provides the clustering capability that ensures high availability of critical resources such as DB2 data, applications, and services. It provides monitoring, failover, and failback capabilities to Heartbeat-defined resources.

5.8.2 Availability characteristics

The availability characteristics are as follows:

- ▶ No planned service outages necessary with parallel sysplex, except for some HW upgrades.
- ▶ If all LPARs are in one server, then some HW upgrades require downtime.
- ▶ Option: use two System z servers for even higher availability and complete freedom from planned service outages.
- ▶ Multiple application images protect from a single application point of failure.

5.8.3 Functionality

- ▶ Dual LPAR in a single site.
- ▶ Single Disk Storage Subsystem is single point of failure.
- ▶ Heartbeat manages the IP address, storage volumes, and DB2.
- ▶ Linux Virtual Server provides the load balancing capability.

5.8.4 Availability scenario

Here is the sequence of events for an unplanned LPAR1 outage:

1. DB2 is running on LPAR1 with the database on a shared file system.
2. LPAR1 experiences a problem due to operator error and ends abnormally.
3. Heartbeat activates the IP address, the volume group, mounts the file system and starts DB2 on the Linux instance on LPAR2.
4. Assuming the two LPARs were sharing 10 processors, the remaining application instance on LPAR2 would still have all 10 processors available.

5.9 Linux on z - GDPS

The model discussed in this section, shown in Figure 5-8 on page 144, is the second option for achieving high availability in a Linux on System z environment.

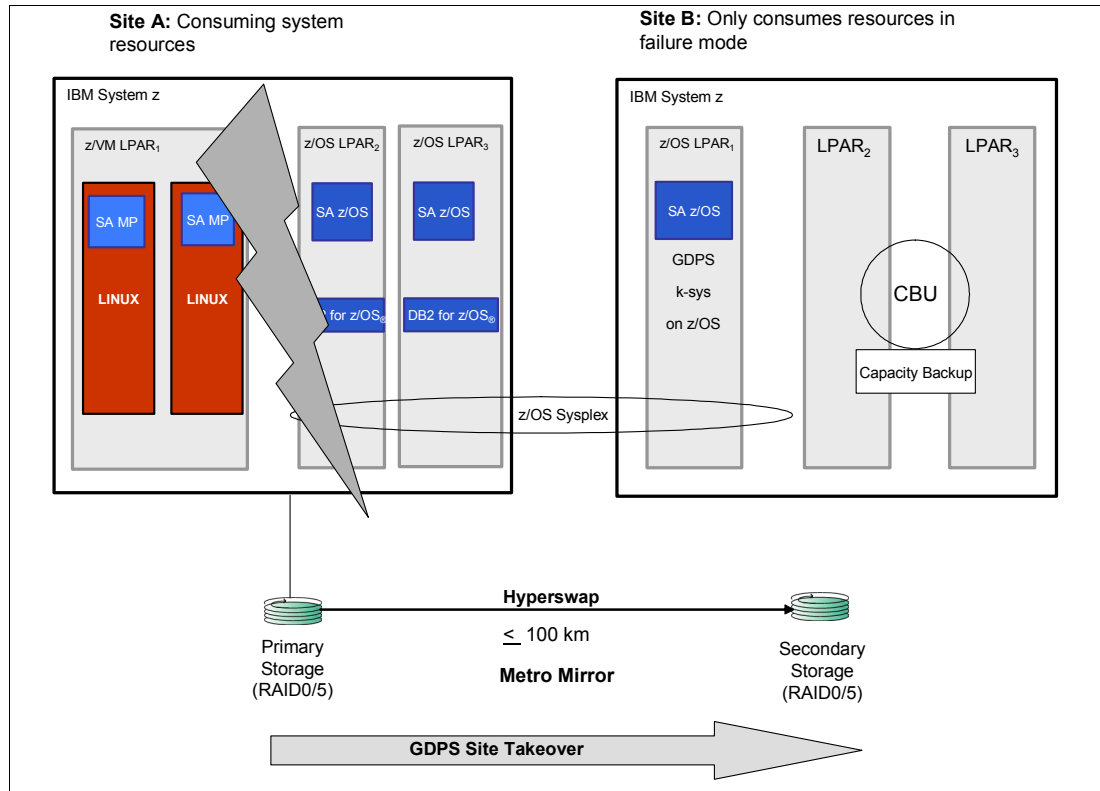


Figure 5-8 Linux on System z - GDPS

5.9.1 Model description

The System z servers in this model are located at two sites with a distance of less than 100 km. This is a combination of a z/VM LPAR hosting two Linux images and a Parallel Sysplex environment with each System z server running two z/OS LPARs on Site A clustered with a GDPS controller LPAR on Site B. There are two disk subsystems. This model employs disk replication with Peer to Peer Remote Copy (PPRC). Hyperswap is used to transparently switch the PPRC primary with the PPRC secondary disk subsystems. Capacity Backup (CBU) is used to provide the additional temporary processor and memory resources at Site B for a disaster scenario.

IBM Tivoli System Automation for z/OS (SA z/OS) monitors the functioning of z/OS and the respective applications and can restart them either in place or on another system in the same or different site.

IBM Tivoli System Automation Multiplatform (SA MP) runs in Linux for System z and on z/VM. This monitors the functioning of z/VM, Linux, and applications and can restart them, either in place or on another system in the same or different site.

5.9.2 Availability characteristics

The availability characteristics are:

- ▶ All work is normally handled by the primary site.
- ▶ No planned outages necessary, except for some HW upgrades.

Option: use two System z servers for even higher availability and complete freedom from planned service outages.

- ▶ Multiple application images protect from a single application point of failure.
- ▶ Disaster recovery based on active/stand-by.

5.9.3 Functionality

- ▶ Dual LPAR; dual site model.
- ▶ Metro Mirror provides remote data copy at the disk subsystem layer.
- ▶ Less than 100 km distance between sites.

5.9.4 Availability scenario

Here is the sequence of events for an unplanned Site A outage:

1. The GDPS K1 System in Site B (using Parallel Sysplex communications) detects this outage and automates the failover process:
 - a. Invoke HyperSwap to switch the surviving z/OS and z/VM systems at the secondary site to use the secondary disks that contain the mirrored data.
 - b. Switch network connectivity to the Site 2 router, based on customer-provided scripts.
 - c. Invoke Capacity Backup (CBU) to increase the number of IFLs available to the VM LPAR and the number of processors available to the z/OS partition.

Note: IFLs, processors and memory cannot be shared between the servers, so if it is required that failover take place immediately and without any interruption to the users, then the server at Site B must have sufficient IFLs, processors and memory to run 100% of the workload. It will take approximately 5 minutes to bring the new IFLs online to z/VM.

2. DB2 will be started by SA z/OS on z/OS LPAR1 in Site 2.
3. z/VM partitions and Linux guests will be started on LPAR2 in Site 2 by SA MP.

5.10 Model summary

Table 5-1 provides a summary of the high availability characteristics of the models discussed in this chapter.

Table 5-1 Summary of the high availability operational models

Model	Sites	Storage Subsystems	Clustering	Disk Mirroring	Data Replication
Single-site Parallel Sysplex	1	1	Virtual	No	No
Dual-site w/data replication	2	2	Physical	No	Yes
Single-site two server Parallel Sysplex	1	1	Physical/Virtual	No	No

Model	Sites	Storage Subsystems	Clustering	Disk Mirroring	Data Replication
Dual-site Parallel Sysplex - Metro Mirror	2	2	Physical/Virtual	Yes	No
Dual-site Parallel Sysplex - Global Mirror	2	2	Physical/Virtual	Yes	No
Linux on z - Linux HA	1	1	Physical/Virtual	No	No

Table 5-2 provides an overview of the disaster recovery characteristics of the operational models that provide disaster recovery.

Table 5-2 Summary of the disaster recovery operational models

Model	Continuous Availability	Disaster Recovery/Metro Distance	Disaster Recovery/Extended Distance
Dual-site w/data replication	X	X	X
Dual-site Parallel Sysplex - Metro Mirror	X	X	
Dual-site Parallel Sysplex - Global Mirror	X	X	X
Linux on z - GDPS	X	X	



Workload assessment

6.1 Workload assessment overview

In this chapter we discuss the important task of assessing an existing application or business system before planning its actual transition to the IBM System z platform. The assessment covers many aspects of business application systems, and introduces a methodology to help identify problem areas early in the transition project.

We use a concept called *Domains of Transformation* to help you focus on aspects of application design or implementation that might make a transition to System z difficult.

Finally, we describe how to fill out a *transformation matrix*, which contains a complete review of the current application with broad effort and risk estimates for each domain.

6.1.1 Workload tiers

Modern business applications are often designed and implemented as separate logical layers or *tiers*. Many modern applications are designed and constructed using a 3-tier approach, as shown in Figure 6-1. Using this approach, an application is designed and constructed as a *presentation tier*, an *application or business logic tier*, and a *data base or file system tier*.

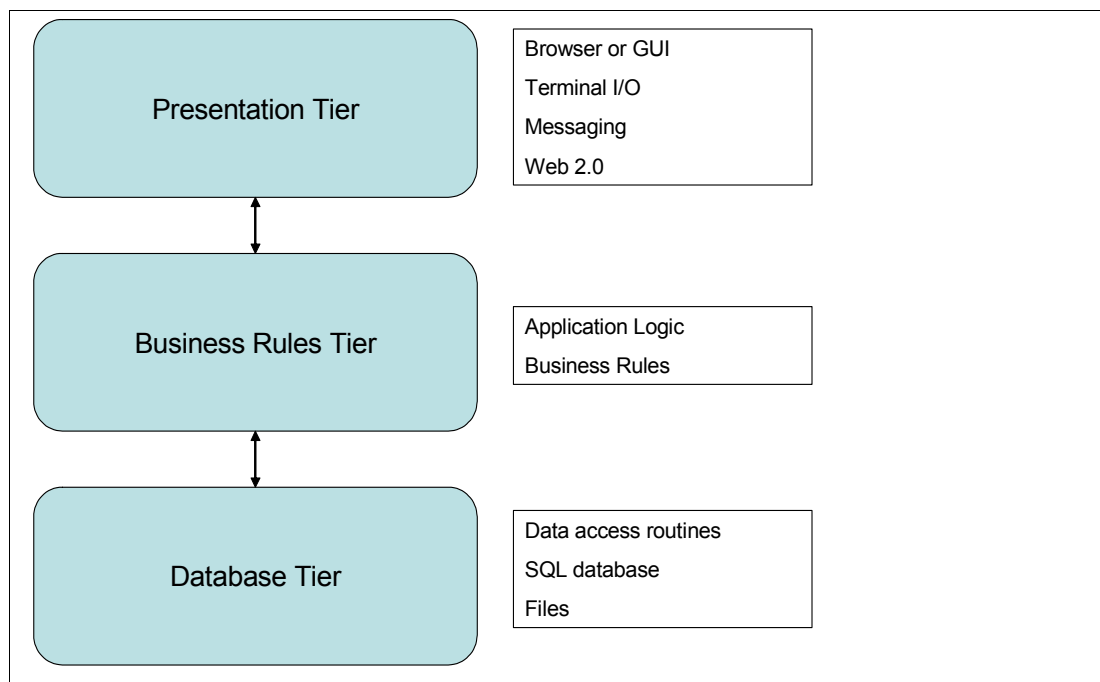


Figure 6-1 Basic 3-tier architecture

Although an older application may not be physically represented as distinct tiers, it is nevertheless useful to think about how the application can be logically decomposed this way, because this may influence your target solution architecture, especially if you cannot just redeploy the application but have to go through a complex migration including recoding.

6.1.2 Domains of Transformation

We now introduce a method to assess and decompose the aspects of an existing application system that will need to change. The goal is to migrate the existing application, which we will refer to by “As Is”, to a new incarnation hosted on another platform; we refer to that

environment as “To Be”. To be very clear, the work ahead is to lay the ground work for moving an application from one hardware and operating system platform, to IBM System z. The work begins by identifying *Domains of Transformation* that, in some sense, represent the personality of the workload. Domains of Transformation differ from simple *characteristics* in that they imply actionability. The work ahead is to define where and how much action is to take place. A transformation domain is a focal point for transformation work.

A characteristic may *not always* imply work to be done, while a transformation domain *always* implies work to be done.

Transforming software does not mean to develop an exact replica of some existing software solution. It does mean to develop a solution that provides the same functionality, with the same look and feel as the initial version.

Every application is composed of key components that exhibit certain qualities. These qualities can be viewed as the external appearance of the application, what it looks like, how it behaves, and how it communicates with the outside world. Some of the qualities of service lend functionality that translates to nonfunctional requirements. A convenient way of approaching this task is to look at the application in terms of its constituent Domains of Transformation, which are the areas of the application that must be transformed into a new form in order to properly function on the new platform. The following list contains the Domains of Transformation that we recognized. However, every environment is different, and thus the list cannot ever be complete for every specific case. The Domains of Transformation matrix should be expanded or modified by the architect team to completely cover all the As Is Domains of Transformation:

- ▶ Application language
Can be nonproprietary, such as COBOL, C, Java, or PL/I; or proprietary, such as SCREEN COBOL, TAL, 3725 Assembler, or DPCX Macro.
- ▶ Application architecture
The architectural *shape* of the application. Is the application cleanly divided into tiers? If so, which tiers need to be transitioned?
- ▶ Proprietary language APIs
These are additions to an otherwise standard language (for example COBOL) that are particular to a specific vendor platform. These may take the form of “calls” to special procedures or APIs that significantly influence the logical structure of programs. Good examples are IBM CICS calls such as EXEC CICS READ and HP PATHWAY calls such as PATHSEND. These calls are special in that they are specific to a vendor’s platform and not the same ANSI standard SQL calls.
- ▶ Availability
How does the application participate in provisioning availability? Multiple models exist that may require programmer intervention such as compiler directives, and so on. A good example would be a system that uses a distribution of process copies across multiple processors in order to provide availability.
- ▶ Scalability
How does the application scale out? There are multiple methods, multiple copies, reentrant code, multiple distributed instances and so on. This is a good time to identify the impediments to scalability, particularly items that may force serialization, I/O bottlenecks (single instance sequential log files), and non-context free transactions.
- ▶ Integration
How does the application relate to other applications, services or processes within the enterprise?

► **Middleware**

What middleware does the application use or depend on?

– **Connectivity**

Connectivity middleware is used to connect different applications to each other, or to their clients. Common examples are IBM WebSphere MQ, Sockets, and APPC.

– **Transaction control and monitoring**

Transaction monitors route transactions to various program resources. Examples of transaction monitors are HP's Pathway(c), the IBM CICS, and BEA Tuxedo.

In a client-server environment, the transactions have to be managed as they transit from the client to the server side code.

► **Security**

What are the security needs of the application? Are there certain techniques being used in the application for auditing, identifying propagation or encryption, for example?

► **Systems management**

Does the application have built-in management features, such as writing event logs?

► **Capacity**

Application systems require system resources to execute. How much memory, processor capacity, DASD and network bandwidth is needed to support the application?

This is a quantitative Domain of Transformation.

► **Transaction framing and integrity**

Transactions may require framing in order to preserve database consistency. A good example is two-phase commit implementations typically framed between a "begin transaction" and "commit", or an "abort" or "rollback" style command. These structures are often used to implement complex program logic, not simply employed to deal with errors.

► **External data structures**

What kind of datastores does the application utilize? This could be an RDBMS, or simply a set of flat files. How is data accessed? Directly, or through connectivity layers such as ODBC?

To what degree does the application employ proprietary functionality in its data access strategy?

► **Proprietary memory management**

How does the application organize its internal memory structures? Does the application use special, or proprietary memory management techniques, or share memory segments between process instances? Does the application do memory management that is specific to a certain programming language, for example?

► **Acquiring transactions**

There are generally two methods for acquiring transactions: user interactive or message processing.

– **User interactive**

Interactive transactions are acquired via a terminal of some sort, processed, and a formulated response returned to the terminal device.

– **Message processing**

Message processing implies a multitiered design and is a common paradigm employed for OLTP. In this case, transactions are communicated to the application as messages

between processes and handled, with the application process returning a response to the calling process. This is often referred to as requestor-server architecture. The requestor process formulates a transaction and sends the transaction to the server process. The server process does some work on the transaction and then returns a response to the requestor process. The requestor-server architecture is similar in function to client-server, except that both client and server are simply processes running on the same, or different, machines.

These Domains of Transformation form the functional primitive description of the workload. Using this description data it is rather straightforward to begin mapping these Domains of Transformation into requirements for the target platform, in this case System z. This activity is the first step in the transformation process. It is from this mapping exercise that the target transformation begins to take shape.

The contents of each Domain of Transformation exist to meet requirements. The application workload takes action with the domain in order to meet a requirement, something that must happen for the application to function correctly.

Domains of Transformation are adaptable. The previous list is a good starting point, but each project may indicate additional Domains of Transformation. Additionally, Domains of Transformation can be layered, or leveled. Within each transformation domain, there can be subdomains, where transformations are more specific. For example, consider the “Availability” domain. Within this domain there could be subdomains, such as:

Process state preservation	Maintaining process pointers for next instruction and data stack contents.
System state preservation	Returning the system to its original configuration after problem remediation.
Failure detection	Determination that a failure has occurred quickly enough to do something about it.

A major success factor is the accurate determination of the relevant Domains of Transformation involved in any specific migration. Resolving how the new system will handle state preservation, or even whether state preservation is necessary on the new platform is the primary work of the migration architecture team.

The following items are characteristics; however, they are not usually a Transformation Domain:

- Batch or OLTP

How does the process execute? There are two basic classes, Online Transaction Processing (OLTP) and Batch processing. OLTP applications are response-time sensitive; there are users, or processes waiting on an immediate response. Batch processes are throughput sensitive. The goal of OLTP processes is to complete transactions as quickly as possible with the shortest possible response time. Batch processes have a totally different goal: to complete as many transactions as possible within some fixed period of time.

- Prerequisites

Some applications have prerequisites that must be in place before the application workload can execute. These prerequisites may be temporal, tied to the timing of external events (end-of-month processing requires that end-of-month has occurred), or process oriented, such as data sorting, or completion of prior predecessor processes (Program A must complete before Program B).

- Compute-or-I/O-ness

Applications generally fall into three categories:

- Business application

Even blend of compute and I/O activity. These applications tend to favor I/O activity.

- Scientific application

Intensive processor workload for math processing.

- I/O bound

Data movement, utility functions and very light logic online transaction processing. This application species tends to dominate the migration landscape.

“Compute-or-I/O-ness” can be viewed as a continuum that spans from pure processor activity (calculate Pi), to pure I/O (copy file-A). This information is important in platform planning. Some processors do high compute work better than others. Constructing a disk subsystem in anticipation of high disk I/O utilization is critical to building a system that does not just work, but that also performs well.

6.1.3 Availability and scalability assessment

This section introduces the concepts of availability and scalability in the assessment phase of the “AS IS” environment. The two topics are brought together by their close architectural relationship: inherently, designs that scale well, are usually highly available, and vice versa. High availability is often a consequence of high scalability.

Scalable environments are able to grow in near, or perfect, linear fashion with the addition of processing resources. In the assessment phase, there are application and infrastructure design features that work against scalability—they will also usually work against availability as well. The fastest way to determine if an application or system is scalable is to search out and identify anti-patterns to scalability and availability. This is almost to say “what is present in, or missing from, our design that will get in the way”.

We will explore further five classes of obstacles that work against both scalability and availability:

- ▶ Exclusively monolithic design - Yields only vertical scalability. Ever larger processors.
- ▶ Exclusive use bottlenecks - Causing contention for data, devices, memory or processes.
- ▶ Increasing synchronization traffic - Growing the architecture causes increases in inter-node traffic. The increase in traffic serves to interfere with response-times as the traffic volumes grow.
- ▶ Process path serialization - Batch processing windows and processing cut off periods, cardinality requirements for transactions, and designs that single thread processing work to inhibit scalability.
- ▶ Failure to follow ACID principals - Atomicity, consistency, isolation and durability. These features comprise the basics of scalability and reliability for an application.

The major warning sign of an application not scaling will be a statement by the system owner to the effect “We had performance problems, we added processors; and, it has not gotten any better”.

Scalability impacts price-performance. Effective utilization of resources implies good scalability. Consider that for the most part, a single thread of a business (I/O involved) application typically cannot drive a single processor to more than about 20% busy. This is because a processor can only exist in one of two states: running or waiting. Running means burning the processor, waiting means just that: waiting for a number of reasons, such as I/O completion or user think time. The fact remains, when a processor is waiting, the processor burn rate drops to zero. It is said that all processors “wait at the same speed”.

Exclusively monolithic design that yields only vertical scalability

Some applications have been designed as monoliths, and these do not have the ability to scale out horizontally due mostly to their design. Applications that cannot scale typically expect an exclusive lock on their databases. This means that only one instance of the application can run at a time, precluding horizontal scaling. They are locked into vertical scale, which is necessarily limited by hardware. These types of applications are sometimes amenable to remediation that allows them to be divided into multiple instances. Other times, the logic is simply too complex and the application is “stuck,” unable to scale for more horsepower; however, even that has an end. Given enough vertical scaling headroom, even the monolith will stop scaling as a result of the design faults that made it a monolith in the first place.

Exclusive use bottlenecks of devices, data, memory or processes

Applications that do not scale well are almost synonymous with “performance problem” applications. They typically will not scale as a result of some bottleneck where a critical resource is being held for exclusive use, for some period of time by another application instance trying to use the same resource. The resource can be a row in a database, a logical file, a physical device, memory, or even another working process. Once the bottleneck has been encountered, the application stops scaling. Adding hardware, or additional instantiations will not advance load volume, response times for OLTP will worsen, and the application will not advance to higher workloads. The hallmark symptoms are:

- ▶ Worsening response times, and low, or at least less than alarming processor utilization levels.
- ▶ Introducing additional traffic does not cause increased overall processor utilization.
- ▶ Transaction rates do not rise with additional users or traffic to the system.

Exclusive use of data

There are application designs that are logically unable to share a database. Usually these designs do not follow ACID principles, or do not have the facilities to control database consistency with multiple concurrent update-capable processes.

Database or Table Exclusive

An application may place locks on entire tables, or the entire database. Table exclusivity is not limited to simply locking; the exclusive use may be the effective result of contention.

Consider an application that must log all of its transactions. Once the transaction is complete, a single row is written to the end of a single log file. This amounts to a single sequential write. As the transaction rate rises, the single log file becomes a point of contention amongst various instantiations of the application code. This may be contention based on overuse of the physical disk device, or contention to update file header data as each new block is written to the table. Either way, only so many transactions can be written to the single log file at one time and the end result is exclusive, or semi-exclusive use of the resource.

Figure 6-2 shows an example of this antipattern. In this pattern, eventually, all of the process instances will be forced to wait on the log file, either for contention for the disk, or logical I/O contention for the data file. The end result will be an inability to scale beyond a certain point due to transaction path serialization.

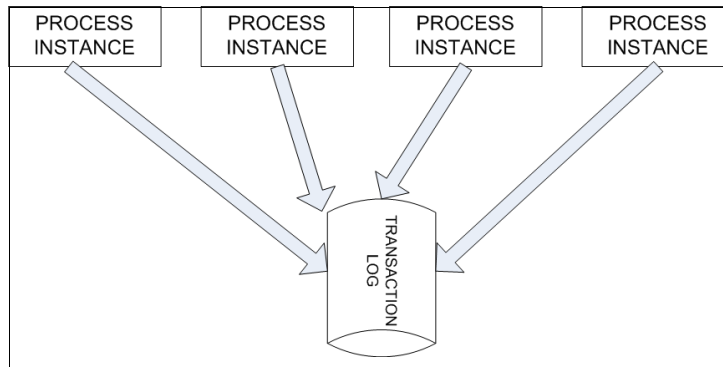


Figure 6-2 Contention in data access

Data row exclusive

All lock contention results in serialization, which negates parallelism. The best designs for locking strategy usually are data row lock centric. Locking the smallest grain of data naturally reduces lock contention; however, there are those who insist on messing up a good thing. Applications that try to lock and update the same row for each transaction simply end up stuck, hanging on an ever growing queue of locks. An example would be a single row in a table that is a transaction counter, where the value in the row is incremented for each transaction. Structures that match this pattern must be changed before the application can grow.

Rapidly increasing synchronization traffic with growth

Designing a system that can scale out horizontally, while a good idea, can run into some pitfalls that will work against scalability. The most significant cause is increasing synchronization traffic required to keep all of the distributed database instances synchronized, so that each has the same data at the same time. This is often referred to as having “a single version of the truth”. Although it is nice to think that you can simply replicate multiple instantiations of a single application to make it grow, updating across communication links of varying speed and reliability will introduce uncertainty into daily operations.

Failure to adhere to ACID principles for transaction processing

The basic “ACID” principles are critical to building scalable and available applications. Reviewing an application for compliance with ACID principles will not only uncover the roadblocks early in the process, but will help to ensure project success. The ACID principles are as follows:

- Atomicity** *Atomicity* implies that all portions of a transaction will complete, or none will. This ensures database consistency. An example would be a system that updates tables in three locations. A compliant application would not be able to leave the three tables in an inconsistent state.
- Consistency** Compliant applications will not leave the database in an inconsistent state regardless of the transaction outcome—success or failure. This means that there will be no data that is invalid, relative to the rules of the database. If a transaction fails, the database will be left in the state previous to the transaction execution. If the transaction is a success, the database is left in a valid state, adhering to all its own rules, ready for the next transaction.
- Isolation** No transaction is aware of any other transaction in progress. Transactions are completely stochastic, in that one transaction does not depend on a previous or subsequent transaction. That means that the sum total of a

transaction is completely contained within the bounds of that transaction. The transaction has no memory, and no impact on the future.

Durability

Once a transaction is completed, it is forever completed, and cannot be undone by a system failure of any type. That means that the transaction does not only exist as changes to the database, but, is preserved physically in a log as well, and preferably in some offsite preservation technique. Durability has a lot in common with the concept of dependability, in that once the application reports a transaction is complete, the owner can depend that the transaction is indeed complete and that the status of that transaction is not going to change.

6.1.4 Integration assessment

No application is an island

When assessing a given workload, the question of how it integrates with other systems and services needs to be considered. Initially there may have been a traditional approach used to integrate with other applications at both the transport and application level.

It is very likely that a distributed application developed in the 1990s will have used techniques such as Remote Procedure Calls (RPC) to integrate with other applications. RPC was an approach popularized by the Distributed Computing Environment (DCE), which was one of the foundations of client-server computing.

There are of course other techniques such as LU 6.2, which was part of the SNA stack and allowed interconnectivity with other systems in the network. Another SNA-based technique was *Advanced Program to Program Communication (APPC)*.

We still find many environments using these approaches. In the modern services-oriented architecture world more and more environments are adopting a message-based architecture to overcome the multivendor integration challenges that exist. The IBM WebSphere MQ is a classic example of this and has become the standard for application integration in many environments.

You should assume that there are interdependencies with other applications or services either within or external to the application environment in the scope. An application that has been operational for many years will in most cases have established interfaces with external applications and used integration techniques that were available at the time and that were compatible.

Many legacy applications have been based on a model where the application owns the data and therefore other applications that need to access that data will have developed techniques to extract and use the data in a way that does not compromise integrity.

In Table 6-1 on page 156 we outline examples of integration that need to be determined and evaluated.

Table 6-1 Integration assessment

Integration requirement	Integration approach	Examples
Access to external services in order to complete the authorization process of a business transaction	<p>Most service providers will have clearly defined standards of communication to their clients and users. One clear example will be credit card schemes.</p> <p>You will have to comply with the communication standards and will most likely have to apply changes on a regular basis to maintain compliance.</p>	<p>SWIFT provides a global financial messaging network for banks and other financial institutions.</p> <p>From 2001 to 2005 they migrated from a x.25 network protocol to IP and established XML as the standard for exchanging messages.</p>
Integrating with back office core functions	<p>Back office systems are often run by separate teams and have operational standards for interconnectivity with peripheral systems and applications.</p> <p>There will be various approaches applied here based on whether synchronous techniques, such as APPC, are adopted.</p> <p>We have approaches such as Remote Procedure Calls (synchronous approach) and messaging (synchronous)</p>	<p>Messaging technologies such as WebSphere MQ are used for messaging.</p> <p>Other implementations include LU6.2.</p>
Exporting data to other repositories for analytics	<p>Many key applications will have developed processes to deliver data for ongoing business analytics.</p> <p>Suitable ETL tools (some more basic than others) will have been used to extract and load key data elements into an appropriate repository to provide business users with key metrics.</p>	<p>Forms of ETL tools are used to transform data from source systems to target environments for business analytics. IBM InfoSphere Information Integrator is an example of a product providing this support.</p>

Communications and connectivity assessment

When assessing the connectivity design of an existing workload, the following key questions must be considered:

- ▶ Is the communications protocol proprietary (for example SNA) or standard (for example TCP/IP)?
- ▶ Is the communications code embedded in the application?
- ▶ Does the application use standard middleware or networking APIs?

Modern applications typically contain no networking logic at all; they delegate this work to their middleware framework. Examples of this type of design include WebSphere MQ applications, which communicate with each other using standard WebSphere MQ APIs, or JEE applications that communicate using either JEE messaging or EJB methods.

Older applications may engage in direct network protocol work. Examples of this kind of design include older socket applications, SNA APPC applications, or perhaps SNA 3270 applications that do not run inside CICS or IMS.

6.1.5 Middleware assessment

An application uses middleware to connect itself with other applications, with existing infrastructure, or with users. Middleware is sometimes called the “glue” that binds different applications together. Some middleware provides *interoperability* between an application and other applications, and sometimes provides an integrated view of the available data. *Enterprise Application Integration (EAI)* middleware is often used to provide various *container* services for applications (for example JEE application servers) or messaging services (such as WebSphere MQ), whereas data integration middleware (such as InfoSphere Information Server) typically provides an integrated view of data.

Some technology previously categorized under middleware is now commonly included in the operating system. The BSD sockets API is now directly supported by most modern OSs but could be considered an example of connectivity middleware. Similarly, Advanced Program to Program Communications (APPC) provides a high-level API that allows applications to communicate with each other, but is usually provided by the system software stack; see Figure 6-3.

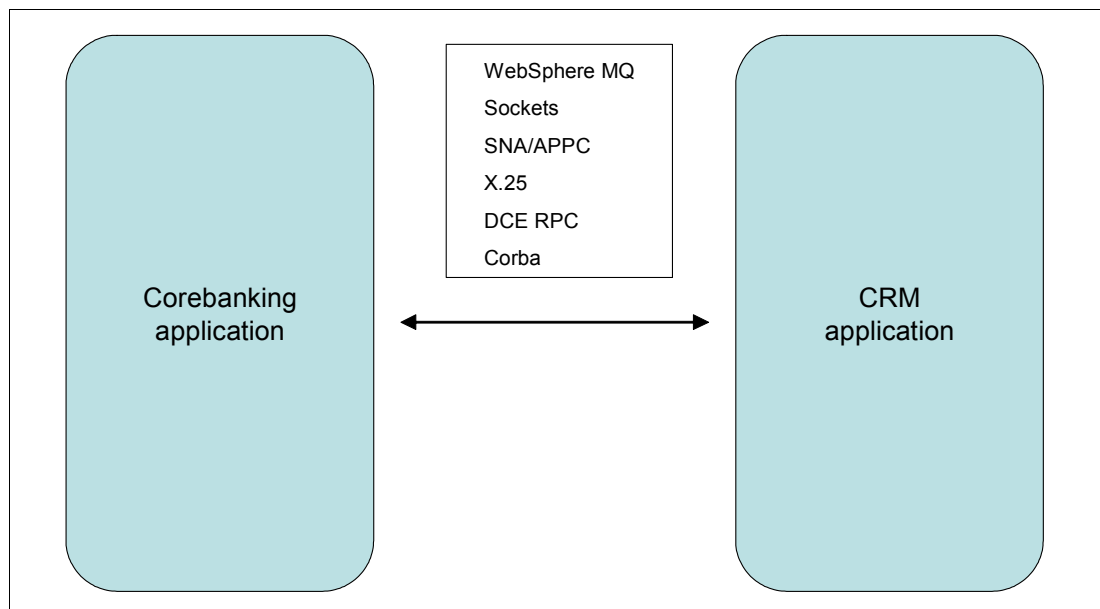


Figure 6-3 Connectivity middleware

Examples of connectivity middleware are:

- ▶ WebSphere MQ
- ▶ Sockets
- ▶ APPC
- ▶ Guardian IPC (on HP NonStop)
- ▶ DCE RPC
- ▶ Corba ORB

These middleware examples are mainly concerned with connecting applications that are dispersed locally or remotely. EAI middleware takes a broader view of the middleware problem by providing a *container* in which the applications run. This kind of middleware may

provide connectivity, but commonly also provides load balancing, transactional semantics, and data integration. Java Platform Enterprise Edition (JEE) application servers are popular examples of this kind of middleware. Within a JEE application server, applications are represented either as Enterprise Java Beans (EJB), Message Data Beans (MDB), or even servlets.

When assessing the impact of middleware on a planned transition to System z, consider:

- ▶ To what degree is the middleware “wired” into the application’s program code?
- ▶ Is the middleware API set or framework available on System z?

Middleware incompatibilities

Before committing to use a middleware solution on System z, you should carefully consider the middleware version that the current application uses. Problems can arise when the application’s code is dependent on a certain middleware version, but that version is not available on System z.

For example, a given Relational Database Management System (RDBMS) might be available on System z, but at RDBMS Version 6.0 in 64-bit mode. The application on the current system is certified and tested only on RDBMS Version 5.0. The application owner (or vendor) may not be able or willing to test and certify the application for Version 6.0 of this RDBMS. The project then stalls, while the architects and engineers determine whether to wait for the application to become certified for Version 6.0, or to choose another middleware solution.

Of all problems that can impede a migration, this area carries the highest potential to delay or even stop a migration project.

Middleware above almost all other components can become a showstopper during a migration effort. The reasons are simple, “application, platform compatibility”. This requires that the architect develop a software compatibility and dependency matrix. Usually this work is watched carefully by the project management team, or project manager. It is not enough to simply understand that some middleware product is available on some platform. Changing operating system versions does not always coincide with matching patch level releases of middleware and database software on a version by version basis. Patches applied to middleware for, say, middleware X, which runs on some hardware platform, may not be in place for the same software on another platform. If there are code dependencies on one of the patches, then the migration effort can grind to a sudden halt when the missing patch is discovered, and endless hours dealing with the software vendors, each pointing fingers at each other until the problem gets resolved.

It is imperative early in the migration planning stages to identify middleware dependencies and patch levels. Ideally, there should be a complete patch log that contains details regarding the patch, its purpose and dependencies. A nonexistent detailed patch log should be considered to be a very high project risk.

Middleware dependencies can have devastating performance implications. A good example is software that is available in both 32-bit and 64-bit versions. The middleware may only be available in a 32-bit version, even though the target (new) platform is a 64-bit environment. Capacity planning that has been used to size the target platform may be adversely impacted so as to cause an unexpected requirement to purchase much more hardware than originally estimated; additionally, response time performance targets may not be achievable using a 32-bit middleware. The end result could be failure to pass acceptance testing.

A good example of this would be an RDBMS, let us call it THEDBMS Version 5.0. It just so happens that THEDBMS is available for the new platform in Version 6.0 in a 64-bit implementation; but, sadly, Version 5.0 is not available on the new platform. The application is

not certified for version 6.0, so now, the project is stalled figuring out how to either get the older version to run on the newer platform, or to certify the application for Version 6.0. Even if the older Version 5.0 would be available, it would run in 32-bit mode, probably giving significantly lower performance.

6.1.6 Database and file systems

The file systems and database management systems employed by applications are generally easy to migrate, with a few exceptions. In this section we examine these.

Here, we draw a distinction between *databases* that typically provide access to data at higher, more abstracted levels, and *file systems* that offer a simpler, more basic, view of data. The distinction is blurred, however; there are proprietary file system types that have features in common with some databases.

Database Management Systems (DBMS) organize data in various ways and offer many services including data creation and removal, and rich collections of data access facilities. DBMSs typically offer sophisticated data abstraction capabilities and usually support local or remote access to that data.

Transactionality

An important common feature of most DBMSs is their support for *transactions*. Transactions are sets of updates to the database that are managed atomically; either an entire set of changes is updated, or none of the set.

Transactions are most useful when different update operations can be managed under a single transaction. A transaction that contains updates from different sources (possibly remote sources) is called a *distributed transaction*. Support for these is commonplace in modern DBMSs. The most common standard for managing distributed transactions is the X/Open XA standard. A DBMS that supports the XA standard can coordinate its work with other transactional systems or products.

The transactional nature of DBMSs is widely exploited by business application systems and must be considered when assessing an existing workload.

Support for transactions is not limited to DBMSs; some simpler file systems also support transactional access to data, although this support is likely to be proprietary rather than based on an open standard such as X/Open XA.

Database management systems

The following summarizes the types of database management systems you may encounter:

- Relational DBMS

Relational databases are typically organized as a set of *tables*, where each table is a collection of rows of similar data. Relational Database Management Systems (RDBMSs) mostly support *Structured Query Language (SQL)* for accessing data. Although the precise details of SQL differ by vendor product, the broad commonality of the query language makes transitioning this type of database easier. Examples of RDBMSs include HP NonStop SQL, IBM DB2, Microsoft SQL Server and Oracle, and the open source MySQL. Most of these products support the XA transaction standard. An exception to this rule is HP NonStop SQL, which supports a proprietary transaction system called TMF.

- Hierarchical DBMS

A hierarchical database organizes each *record* of data into a tree-like structure. The hierarchical tree structure relates information using parent-child relationships: each parent

can have many children but each child only has one parent (also known as a 1:many ratio). The most common example of a hierarchical DBMS is the IBM Information Management System (IMS) product. Like RDBMSs, IMS has a query language called Data Language 1 (DL1), but these days IMS databases can also be accessed using SQL.

- Other DB variants

There are many other types of databases that offer a higher level view of data, or a more specialized view. Examples include object databases that manage access to objects rather than simple records, and operational databases that typically contain specialized data about an organization's operations. These more abstracted and specialized data models can be, and often are, implemented using an RDBMS.

File systems

File systems can be broken down into a few categories:

- Entry-sequenced files

Entry-sequenced files contain records (either fixed or variable length) that exist in the order they were added. Records are typically appended to an entry-sequenced file and applications can only access that record by retaining its *address* in the file. The address might be a relative record number, or bytes offset. Examples of common entry-sequenced files are IBM VSAM ESDSs and RRDs and HP NonStop Enscribe entry-sequenced files.

- Key-sequenced files

Key-sequenced files are accessed using a *key*. Each record in the file has a key that the accessing program uses to locate it. Record keys can be unique within the file or duplicate. The ordering of keys determines the order in which the data records are seen by programs. Many key-sequenced file systems support multiple keys so that records can be viewed in more than one sequence. Examples of common key-sequenced files are IBM VSAM KSDSs and HP NonStop Enscribe key-sequenced files.

- Flat files

So-called *flat* files are simple streams of bytes that have no internal structure imposed by the file system, other than a length. Flat files can contain unstructured binary data or text lines separated by a designated line-ending character. Data can be appended to a flat file or can be written to some location within the file. Unix systems support flat files. Examples of flat files include all UNIX files, VSAM LDSs, and HP NonStop Enscribe unstructured files.

- Distributed file systems

Distributed file systems feature functions that allow the file system to distribute file structures across multiple nodes. Using this method, an application does not have knowledge about the file's physical distribution across devices, or nodes. Examples of this are HP NonStop Expand network, and the IBM General Parallel File System (GPFS™). Generally, these file systems provide facilities to permit highly available scalable systems without adding absurd levels of synchronization overhead. Data may be distributed by key value across nodes in order to balance I/O traffic. This scheme is inflexible and may result in poor performance in the event that I/O traffic suddenly changes, perhaps due to a holiday season. An example would be looking up florists and flowers on Valentine's day. A system with data distributed by key value might exhibit poor performance as everyone would use the same disk volume.

- Journal file systems

Journal file systems allow very rapid recovery of file systems after a crash without requiring to walk through the entire data structure. A journal file system notes the changes it intends to make, ahead of time. After a system failure, the journal file is used to replay transactions and bring the file system up to date. This adheres to the "atomicity"

requirement for ACID design—once the journal log has been replayed, transactions are either complete, or discarded (see “Failure to adhere to ACID principles for transaction processing” on page 154). Not all operating systems support journal file systems; and some such as HP Nonstop have middleware to support this type of activity. Journaling may involve audit trails and application code to handle rollback, rollforward, and abort transactions. The use of these features may not be solely for handling failures, but may be part of the mainstream application logic.

- Inter-process and device files

File systems, aside from providing storage, may also serve as a method of communication between processes and devices. The storage component of the various file systems does not usually pose migration issues; almost all support the following normal flat file systems: relative, indexed, and entry-sequenced files.

6.1.7 Security assessment

Application security is a key feature of any IT system that demands careful consideration during the transition to System z. This section gives a brief overview of the concepts of application security, and how to assess the security attributes of an existing application.

The term application security covers a number of concepts that should be understood and considered when assessing an application system.

Confidentiality

Confidentiality (or privacy) is the assurance that information is not disclosed to unauthorized persons, organizations, processes, or systems. Confidentiality protects sensitive information from disclosure.

IT systems can provide confidentiality for data that is stored locally with its file system, and or for data being moved between systems over a network. In many cases, IT systems provide confidentiality for both locally stored data and data that is being moved over a network.

Confidentiality or privacy can be based purely on authenticated user controls, or the data itself can be encrypted. Data encryption falls into two distinct categories:

- Data in files

Data that is stored in files can be encrypted either at the file level, or at the disk volume level. A number of platforms, including System z, support both types of file encryption.

- Data flowing over a network

Data being transmitted over a network can be encrypted using the now standard Secure Sockets Layer/Transport Level Security (SSL/TLS) protocols. These protocols can use X509 certificates to establish authentication of network endpoints before transmission can proceed.

There are a few aspects that are critical to know when attempting a migration to another platform:

- Is there any encryption functionality embedded in the application code?
For example, in case of a Java application this could be usage of any of the Java security frameworks, such as the *Java Secure Socket Extension (JSSE)* library.
- Are there any ISV packages, frameworks or libraries used from the application to perform encryption?
- Is hardware encryption used on the current platform?

- On inbound encrypted data traffic, what are the specific (DES) standards required on the server side?

Authentication

Authentication (also referred to as identification) is the assurance that a person or system is who or what they say they are.

Authenticating systems will unambiguously identify users and programs, verify those identities, and will assure individual accountability.

The authentication of users is required to ensure that they are associated with the proper security attributes (for example, identity, groups, roles, security, or integrity levels). The unambiguous identification of authorized users and the correct association of security attributes with users and subjects is critical to the enforcement of the intended security policies.

Authentication is usually done in two common ways:

- User ID and password authentication, also known as “basic authentication”. This method is the simplest and is based on the idea that each user has a unique identifying name (user ID) and a secret password. This type of authentication offers very little protection against hackers.
- Certificate-based authentication. Each user has a certificate that uniquely identifies them to the IT system. Both Pretty Good Privacy (PGP) and X509 are examples of common certificate-based authentication systems. This offers better protection.

Authentication is provided by most modern operating systems using either user ID and password tokens or X509 certificates. Middleware products typically rely on the underlying OS to manage authentication, although network authentication can also be done by any product using the Secure Sockets Layer (or TLS).

It is an unusual and generally bad practice to do authentication in the application, but it does happen here and there. You have to keep in mind that security tolerance on System z is more strict than on most other platforms, so a bad practice accepted on your source platform, such as doing authentication in the application, may be unacceptable on System z. If you do perform authentication in the application, you have to be prepared to move that duty to the middleware or the O/S level.

Re-authentication

In some applications authentication is done multiple times between different layers of the application. So, for example, the first authentication is done when a user tries to access a web server and a second authentication is done when accessing a database.

Single-Sign On

In a Single-Sign On (SSO) a user only needs to authenticate once while accessing multiple applications. The idea is that users authenticate and then receive a token that they have been authenticated already. With that token they can continue accessing various applications without being challenged again. Of course, this type of token times out after a set time limit.

Authorization

Once users have been authenticated and been granted access to the application, it does not mean yet that they can do everything and access all data in the application. Further *authorization* checks are many times necessary to allow the user to access specific branches in a program, invoke calls to other programs or access specific sets of data.

Authorization is probably the most common area of security to take care of in the application code. Authorization is many times a function of the application, and an administrator maintains a set of tables that control authorization. This entire structure may be easily portable, especially if the tables are maintained in a common database, such as DB2.

Non-repudiation

Non-repudiation is another important attribute of IT security and can be thought of as an extension to the authentication and identification of users. An IT system that enforces non-repudiation ensures that a transaction or message cannot subsequently be denied by the originating user or party.

Strong authentication is a prerequisite for enforcing non-repudiation, since you must first positively know who the originating user is.

Access control

Access control is a term that refers to the IT system's ability to grant or refuse access to data or services based on the identity of the requesting user. This clearly requires that an effective authentication method is in place. Organizations use access control to protect critical resources by limiting access to only authorized and authenticated users.

Access control is usually provided by the operating system (OS) because it mediates access to most system resources. Middleware products such as CICS, IMS/DC or HP NonStop PATHWAY can also implement access control. OS-based access control is usually based on file permissions and is often extended with *Access Control Lists (ACLs)* that offer finer-grained control of resource access.

Assessing application security

When considering the security characteristics of an existing application system, it is important to identify which components are actually providing the security.

Embedded security

Many older applications may be designed to provide their own security functions. These applications might implement user authentication or data confidentiality directly within their own code.

Older applications may prompt users directly for their name, ID and password and may verify these details against their own, internally maintained, security database or file. Additionally, applications of this type might implement their own data encryption either on locally stored files, or when communicating over a network.

If this is the case, it has to be analyzed whether it is desirable to move over the embedded security functionality as-is and whether the functionality would work the same way it did on the source platform. It may be more desirable to extract the security functionality from the application and delegate it to middleware or the OS.

Infrastructure-provided security

Modern applications do not typically provide security directly, but rather rely on the operation system or other infrastructure and middleware to provide it. All modern operating systems offer some form of password-based user authentication and many provide elaborate resource access controls based on that authentication.

If middleware is used, and this same middleware is available on System z, security functionality may be migrated without complications. However, if this is not the case, or ISV products are used that are not available on System z, then this will complicate the migration.

6.1.8 Systems management assessment

Systems management does not need to become an issue in a migration, as long as the tools are available on the target platform to ensure the agreed SLAs. Nevertheless, we offer a brief discussion on this topic in the following sections.

6.1.9 Capacity

An important part of the workload assessment is the quantitative aspect, meaning how large is the workload in terms of transactions, memory consumption, storage consumption and so on, and how much capacity is currently used to run the workload. When migrating to System z, knowledge is required about these things, to be able to make a proper sizing for System z. In Chapter 8, “System z capacity planning, sizing, and TCO” on page 187 there is a detailed discussion about sizing and capacity planning; we just summarize the information to focus on in the following sections.

Workload volume assessment and forecasting

Systems have to be sized. That is done based on usage and resource consumption. When applications are reengineered, the resources consumed per service request will typically change, but reengineering does not change the number of service requests.

This section focuses on determining the pattern of service requests. Legacy systems, defined as systems already in production, usually have the benefit of recording some aspects of actual usage patterns. Unfortunately, the measures are often used incorrectly for forecasting growth, peaks, and capacity requirements. Data that would help planning for a variable capacity mixed workload system is often not readily available.

We consider several ways of estimating future workloads, based on available information and what is known about systems where service requests come from many sources.

The ideal information to collect

When workloads are consolidated onto mixed workload systems, it is helpful to know the pattern of peaks over days of the week, days of the month, months, and hours of the day. The number of requests varies from second to second, as can be expected when service requests come from a very large number of independent sources. The quantity of requests in any one second is a random variable. The mix of these quantities follows the Poisson statistical distribution.

The starting point for analysis is the average service request arrival rate. Once you know the average, you can calculate the likelihood of any particular peak and the likelihood of exceeding any peak you might consider.

Peak second of the year

Knowing how high a river rose last year does not tell you how high the river can be expected to rise this year. It does not tell you the highest it can be expected to rise in any of the next one hundred years.

Knowing the peak second of the year, in and of itself, is not very useful. It is one measurement of a random variable. The peak is a measurement far on the tail of a distribution. You cannot derive the average from it. It does not tell you whether that peak value had a likelihood of 1/1,000, 1/3,600, 1/100,000 or was surprisingly low, or high. It does not help very much to size a solution.

Service request rate - average over the peak hour of the year

Determine the actual rate of service requests over a peak load span of time when the workload is consistent, such as the peak hour.

Average peak second for each minute in the peak hour

Some usage reporting systems determine the volume in the peak second of every minute and then report the average of those numbers. This is not the same as the average of the highest 60 seconds of the hour. It is also not the same as the median peak second for each minute in the peak hour. At least that would tell you that 30 minutes had higher peaks and 29 had lower peaks.

That does not mean that average peak second of each minute is a useless number. If you have it, you can work backward toward the average rate, and scale up from there. Simulations for rates at and above 100 TPS show that the average peak second for each minute in an hour is between 2.3 and 2.4 standard deviations above the average.

Peak hour on the days with the 5th to 15th busiest peak hours

Does the workload have seasonal peaks such that a few days per year need more capacity than the rest? If it does, then you can save by giving your system a temporary performance boost on those days.

On IBM System z and System p®, unpurchased extra capacity is often physically installed in the servers. It is available for use, priced on a per diem basis. It can be activated manually or automatically, based on rules you set.

Knowing the anticipated usage pattern helps you plan how much to buy and how much to exploit on a per diem basis.

Migration considerations

The number of standard deviations, s , above peak hour TPS to size for is a decision that varies with platform. The peak TPS to size for on a dedicated system is higher than on a shared multiple workload system. That will be discussed in “Sizing for consolidation or migration” on page 213.

Growth limits

Sometimes there are limits on growth. The number of client accounts might be limited by population size. The ATM transaction rate may be limited by the number of ATMs and how long it takes for the next person in line to notice that the previous user is starting to walk away, for this user to walk up, insert a card, do the average business transaction, and start to walk away.

6.2 Completing the assessment

In the previous section, a candidate list of common domains of transformation is listed and described. Those domains represent aspects or features of application systems that often need to change when migrating to System z.

6.2.1 Defining the domains of transformation matrix

The first step is to firm up the Domains of Transformation that are involved in the migration. The list provided in “Domains of Transformation” on page 148 is a good starting place, and for

many applications is adequate. Each application workload may contain additional Domains of Transformation, and so the list can be expanded, domains can be added, or removed.

Once this is done, the Domains of Transformation need to be documented. A good way to accomplish this is with a simple matrix that describes the workload content by transformation domain, possible target transformation possibilities, and then to assign some indication of the work complexity and the project risk involved in accomplishing transformation within each transformation domain. A good way to do this is to assign a relative value to indicate the complexity and risk of each transformation domain. Initially the architect may not know exactly what is involved, or even what transformation paths are available on the target platform. With no established transformation path, both complexity and risk are assumed to be their maximum. Using a simple scale of 1 to 10, 10 being most difficult and most risky, not knowing an exact solution for the domain, we assign 10s for both. Example 6-1 is an example for one of the transformation domains, "Programming language".

Example 6-1 Example for the programming language transformation domain - Initial state

Transformation domain	Domain contents	System z target transformation possibilities	Complexity 0 to 10	Estimated person hours	Risk 0-10
Programming language	C like programming language, proprietary	C, Java, or a combination	10		10

The Top 10 matrix shown in Table 6-2 on page 166 is based upon 10 domains of transformation that have associated with them ratings for estimated difficulty, and risk, the rating from 1 to 10, 10 being the most difficult, or more risky. The hardest possible migration, with the highest risk, would subjectively achieve a score of 10 for both difficulty and risk; as project work progresses, each transformation domain develops a more defined and certain System z solution component that yields reduced uncertainty, and risk.

A fully completed project would have an estimated work difficulty of zero since there is no more work to do, and a risk of zero, since there is no more risk. Risk indicates the risk that the transformation domain can be transformed to a viable System z solution in the estimated hours. In an initial analysis, estimated hours could be set to infinity. The question then becomes: "can this migration be done in a reasonable amount of time, relative to project budget and time constraints?"

Table 6-2 Example Top 10 matrix for a proprietary application workload

Transformation domain	Domain contents	System z target transformation possibilities	Est. work difficulty 0 to 10	Est. person hours	Risk 0-10
Programming language	31 application programs, a proprietary C like language	System z source language possibilities: C migration, some third party tools	6	1240	7
Proprietary language extensions	Program uses proprietary language extensions to send messages from one process to another.	Will require research. Possible fit is WMQ, sockets or other options. This item is open.	8	600	7
Availability	Continuously available, can drop inflight transactions, but must remain up 7x24, uses proprietary scheme composed of hardware and software.	System z option available is Parallel Sysplex.	5	300	5

Transformation domain	Domain contents	System z target transformation possibilities	Est. work difficulty 0 to 10	Est. person hours	Risk 0-10
Scalability	Achieves linear scalability with additional processors and nodes, requires use of proprietary transaction monitor and process management scheme. Application is designed to provide good vertical scaling with features to implement horizontal scalability.	z/OS scalability achieved by using multiple LPARs. Numerous software options. Requires research.	5	24	3
Transaction framing and integrity	Uses proprietary one- and two-phase commit with proprietary language extensions.	Similar schemes available on z/OS; this requires research.	5	160	5
External data structures	Uses SQL-based RDBMS	Similar functions on System z.	3	120	3
Proprietary memory management	Uses proprietary method to share memory between running processes.	Requires research, unknown.	10	300	10
Instrumentation	Process is instrumented with counters that are collected and managed by proprietary software to keep track of system utilization, disk I/O, and use-based counters.	Requires research.	10	300	7
Transaction acquisition	Transactions arrive via IP to inbound process that farms work out to worker processes that accomplish work. Responses are then returned on a per transaction basis.	Requires research. Several options available, such as sockets and WMQ.	5	120	4
Transaction control and monitoring	Handled by proprietary transaction monitor similar to Tuxedo or CICS.	Multiple System z solutions possible. Needs research.	4	120	6
		TOTAL SCORE	64		57

In this example, the top 10 provides a composite score of 64 out of a possible 100 for difficulty, and 57 out of 100 for project risk; this indicates a moderately difficult migration with a significant amount of risk, given the limited information available. As more insight is obtained, these values trend lower.

The evaluation result of the “As Is” assessment translates into a set of requirements and design principles for the “To Be” environment. Every step of the analysis at this phase should be directly targeting “this is what happens, and these are the requirements being met on the As Is platform”, “how do we make that happen and this is how we meet requirements on the To Be System z platform?” Some of these solutions may require significant rework, and along with that significant project risk, to accomplish.

6.2.2 Assigning complexity and risk ratings

These two subjects, risk and complexity, are presented together. They are inseparable and complimentary. Complexity yields risk, risk aversion may in itself increase complexity.

Each Domain of Transformation is assigned a work complexity, and risk rating using a simple scale of one to ten, 0 being the simplest, requiring no work, and 10, requiring extensive redesign or software development in order to meet functionality. Risk is assigned a similar value of 0 to 10, which rates the possibility that whatever work is to be accomplished, cannot be completed within project time and or budget guidelines.

In the early phases of project definition, the exact number of work hours to complete transformation of a specific domain is not yet understood, simply because no target System z solution has been selected. A good example might be a recompile of a simple COBOL program from one UNIX implementation to another UNIX implementation, which would probably not require much work to transform source code, and the path is well understood, and there is little complexity.

Initially when there is no clear transformation path identified, the risk and complexity rating might be close to 10, especially if there are no published tools to handle the source conversion. This activity is first accomplished with incomplete data, because only possible System z transformations have been identified, there has been no firm decision, and possibly very little research. Completing this activity early on will provide good guidance regarding where to focus efforts. In the initial stages of a migration, where very little is known and few decisions have been made, the project may appear from the perspective of the matrix to be very complex and very risky.

The early assessment ratings are not perfect; for some Domains of Transformation, there may not be an identified direct corollary on the target platform, meaning that there may be a substantial build activity, or possibly massive restructure of the application code, or perhaps very little known prior art examples—a project may be living on what is known as the “bleeding edge”.

One of the goals of any migration plan is to arrive quickly at a go/nogo decision. Utilizing data from the matrix that has not been given a chance to develop could result in a premature nogo decision, based on an inflated perception of project risk and complexity.

Factors that impact risk and complexity

The risk associated with the completion of any Domain of Transformation is tightly correlated with the following factors:

- The degree to which the transformation domain is understood.

Is there any evidence of similar migrations done successfully? Something which is done routinely throughout the industry would not pose a significant risk compared to work that involves living on the bleeding edge.

- Transformation expertise required.

The seniority, skill level and number of subject matter experts required to accomplish the transformation. A task that can be handled by a single programmer, that requires no support, or prerequisite setup by others would not be a complex task. On the other hand, installation and configuration of a large enterprise system with many subsystems and complex software setup requiring the work of several subject matter experts would be a complex Domain of Transformation. Transforming a complete bisync network for ATM machines to all IPs would require change across multiple Domains of Transformation and would be considered complex but not high risk, since the path is clear and the personnel are available.

- ▶ The certainty of a specific solution path to meet the requirements of the transformation.
For instance, a COBOL program to be transformed to a platform with a similar COBOL program involves less complexity and less risk than transformation of Java code to a platform that does not have a Java compiler and never will. Fewer instances of any domain having been transformed before by others, poor outcomes on previous projects, and the need to develop special tools that do not exist all work to raise both complexity and risk. Increases in the count of transformations only translates into increased hours, but not complexity. Copying 200 programs is some amount of work, but no matter how many are moved, this does not really increase the complexity of the task.
- ▶ The degree to which a transformation path is decided upon further changes the risk involved, and possibly the complexity.
- ▶ Time or budget constraints may work to increase risk relative to expectations. Operating on an inadequate budget of time or funding can lead to early or poor decision making, or avoidance of what may appear as longer-range options. Inadequate time or budget to complete can doom a transformation before it starts.
- ▶ Availability of business subject matter experts.
Transformations must function correctly relative to user expectations. Business rules and potentially governmental rules and regulations can have an impact on design decisions and require attention early during transformation efforts.
- ▶ Availability of technology resources.
Hardware, software, connectivity, and development tools will impact time to complete. It is imperative that all physical and software resources be available on schedule with the transformation project.
- ▶ Performance expectations and measurements are one of the most common causes of transformation failures. Performance expectations for each component must be developed, and new transformed components tested as early as possible in the transformation work. Many projects have failed after a product that meets all of the functional requirements fails to meet one or more nonfunctional requirements—the most common, performance.

6.3 Workload assessment summary

In this chapter we discussed how to qualitatively assess an application workload as it exists on the current system. The goals of the workload assessment are to determine the effort and risk associated with transitioning the application to System z. Specifically, the goals of the workload assessment are:

- ▶ Identify aspects or features of the application that might need to change.
- ▶ For each identified change, assess the difficulty and risk of implementing the change.
- ▶ Identify potential problems or road blocks to transitioning the application to System z.

The assessment is driven by identifying *Domains of Transformation*, which defines and focuses effort on aspects of the application that will likely require redesign or reimplementing during the transition.

In this chapter we deal with a set of commonly encountered Domains of Transformation, and give an overview of each.

Finally, we describe how to create a Domain of Transformation matrix that contains the effort and risk associated with each change. This matrix is used when designing and planning the work to transition an application.



Transition approaches

This chapter provides a framework by which you can plan to move your existing application to System z. It provides patterns or models that describe typical applications and transitions. These patterns help you make planning decisions for your application. Additionally, the chapter explains ways in which you can switch over from your old application to the new one on System z.

The goal of the information in this chapter is to enable you to determine how complex a migration will turn out to be, based on the current application and SLA.

7.1 Introduction and overview of transitional approaches

Planning for your application transition includes determining the characteristics of your existing application (the application pattern), the strategy by which you transition the application to System z (the transitional pattern), and, finally, the switchover from the existing application to the new one on System z.

Consider *application patterns* as typical models of applications. The attempt is to provide common application patterns that help you think about the characteristics of your existing application, which likely resembles one application pattern more than others. The application patterns discussed are:

Application pattern 1 (AP1)	An application that uses a binary-compatible programming language
Application pattern 2 (AP2)	An application that is written in a standard language supported on System z
Application pattern 3 (AP3)	An application with a proprietary language and data store

Each application pattern maps to one or more transitional patterns. A *transitional pattern* is the strategy by which an existing application is transitioned to System z. The transitional patterns are:

Transitional pattern 1 (TP1)	Buy a software package (that is, replace the existing application) or recode the application from scratch
Transitional pattern 2 (TP2)	Redeploy the existing application through code conversion tools
Transitional pattern 3 (TP3)	Port the existing application recompiling on System z
Transitional pattern 4 (TP4)	Redeploy the existing application as it exists today

A final decision in your transitional plan is how to switch over to the new application. You can switch over in the following ways:

- ▶ By using the “big bang” approach, in which the old application is shut down and the new application is started immediately after shutdown.
- ▶ By using the “shadowing” approach, in which both the existing application on the original platform and the new application on System z run in parallel for a period of time. This parallelism can be achieved in various ways, such as by copying data or mirroring from one application to the other, or by simply feeding both application versions the same input to process.
- ▶ By using the phased approach, in which parts of the application are transitioned while other parts remain on the existing system.

Note the following assumptions made in this chapter:

- ▶ Your System z target environment exists in your enterprise and your enterprise has the required skills, although perhaps not all System z functions and facilities that you might require are enabled. For instance, you have a System z system in your enterprise, but the system is not sysplex enabled. For more information about the functions and facilities that System z offers, see Chapter 4, “System z technology options” on page 49.
- ▶ The System z target environment has sufficient capacity for your application. For more information about sizing and capacity planning, see Chapter 8, “System z capacity planning, sizing, and TCO” on page 187.
- ▶ The focus is on the existing application itself, not on the infrastructure or middleware.

7.2 Application patterns

The application patterns exemplify the degree of difficulty in transitioning an application to System z. You can judge the degree of difficulty of your application by how well it fits one of these patterns.

7.2.1 AP1: An application using a binary-compatible language

This pattern is the easiest to transition. The existing application is written in a platform-independent language such as Java and uses platform-independent middleware, such as WebSphere Application Server. As a database, the application uses a database accessible by SQL.

Important: Although SQL is a common standard for specifying database definitions and manipulating database content (insert, change and delete rows of data), the exact syntax of SQL statements may differ between databases and platforms. Ideally, your application has a separate data layer that makes it easy to adjust the syntax of the SQL statements to the target database and platform.

For example, the System z target environment uses WebSphere Application Server as the middleware, which provides the same Java containers as the existing application, so the business logic does not have to change. The presentation layer is provided by Web browsers, so it does not have to change either. DB2 for z/OS provides the database, requiring an export of the relational data from the existing system to the target environment.

7.2.2 AP2: An application written in a standard language supported on System z

This pattern uses standard languages and data stores, but each platform has its own version, thus requiring more work for the transition. Programming languages in this category include C/C++ and COBOL. Actually, SQL also falls into this category, because SQL is not exactly the same on all platforms, even when using the same database product.

Some changes may be required in the SQL statements. In addition, if the application is an online transaction processing application, the middleware might need to be changed or replaced, depending on which container you use on System z. The same might be true for the presentation layer. At the very minimum you will need to recompile and retest all the code.

You might also consider opportunities for modernization during this transition. In case you decide to switch to other middleware, you might be able to leverage additional functions offered by the new middleware.

7.2.3 AP3: An application with a proprietary language and data store

This pattern is the most difficult transition, because all aspects of the application must be replaced:

- ▶ The business logic must be rewritten in a new language that is supported on System z.
- ▶ If an application has a user interface, the presentation layer must be replaced.
- ▶ If a proprietary file system is used by the existing application, it must be replaced by an equivalent file system on System z.

- With this type of application, consider modernizing the application to use middleware such as WebSphere Application Server and a standards-based RDBMS such as DB2.

7.2.4 Summary of application patterns

Table 7-1 maps the transitional domains from Chapter 6, “Workload assessment” on page 147 to the application patterns. This information is useful when you decide which transitional pattern you follow.

Table 7-1 Mapping of transitional domains to application patterns¹

Domain of change	AP1: binary-compatible language	AP2: standard language	AP3: proprietary language
Application language	-	Recompile	Recode, convert, or replace with a vendor package
Application architecture <ul style="list-style-type: none"> ► Transaction framing and integrity ► External data structures ► Proprietary memory management ► Instrumentation 	-	Unless standard middleware is used, change middleware-dependent parts of the application	Depending on the existing architecture, might change or remain intact.
Proprietary language APIs	-	Reimplement using equivalent System z APIs	Reimplement using equivalent System z APIs
Availability	Depends on System z configuration and capacity	Portions might need to be reimplemented using System z availability features	Portions might need to be reimplemented using System z availability features
Scalability	Depends on System z configuration and capacity	Portions might need to be implemented to exploit System z scalability features	Portions might need to be implemented to exploit System z scalability features
Integration	Connectivity to related applications must be preserved or provided	Connectivity to related applications must be preserved or provided	Connectivity to related applications must be preserved or provided
Middleware <ul style="list-style-type: none"> ► Acquiring transactions ► Transaction control and monitoring 	Changes to available System z middleware	Middleware-dependent portions might need to be reimplemented	Middleware-dependent portions must be reimplemented
Security	Provided by available System z security software	Application-managed security can be ported or modernized using System z security software	Application-managed security can be ported or modernized using System z security software

¹ If there is nothing to do for a transitional domain, the table cell contains a dash (-).

Domain of change	AP1: binary-compatible language	AP2: standard language	AP3: proprietary language
Systems management	Provided by available System z management software. Might be able to reuse existing management software?	Provided by available System z management software. Might be able to reuse existing management software?	Provided by available System z management software. Might be able to reuse existing management software?
Capacity and sizing	Must recalibrate for System z	Must recalibrate for System z	Must recalibrate for System z

7.3 Transitional patterns

The following sections describe typical transitional patterns according to their focus or ultimate goal, and their characteristics.

7.3.1 TP1: Buy a software package or recode the application from scratch

The focus for this transitional pattern is to gain *new* function. Because the existing application must be replaced through either buying a vendor software package or recoding, the primary motivator for this transitional pattern is to gain new function while making the transition. The functionality of the existing application must be preserved, but because you are replacing the application, a key benefit is to gain new functions that can be exploited in the future.

Another aspect of the transition is the application architecture, that is, how well the application tiers are separated. Some applications are essentially one mass of code, with presentation, business logic, and middleware/database functions intertwined. Other applications have clearer distinctions between the presentation layer, business logic, and middleware or database.

- ▶ For applications coded with a proprietary language, the presentation layer probably needs to be recoded on System z. For instance, you will need to recode the terminal I/O for UNIX-style applications.
- ▶ If the business logic for the existing application is separate from the presentation layer, this part might be ported to System z relatively easily if the business logic is coded in a standard language such as C/C++.
- ▶ The database can be replaced by DB2 on System z, in which case you will need to export the data from your old application to the new one.

7.3.2 TP2: source code conversion

The focus for this transitional pattern is *reuse*. You are satisfied with the functionality of the existing application and want to preserve the functions as they are today. The task is to employ conversion tools that preserve your current application's functions.

This transition pattern is centered on translating or converting the source code of the current application into a language that is available on System z. As with translating a book or manuscript into a different language, the intent is to preserve the meaning of the old, in the new language.

When performed manually, source code conversion can be error-prone and it is a highly labor-intensive process. Manually converting an application's source code to a new language is usually justifiable for small application systems. In this discussion, "small" is related to the number of source modules in the application and their complexity.

For applications with more than a few source modules and moderately complex design, it is more cost effective to use a source code conversion tool to at least assist with the conversion.

7.3.3 TP3: Port the existing application by recompiling on System z

The focus for this transitional pattern is to minimize *risk*. Because your existing application is in a standard language, you can minimize the risk by recompiling the application on System z.

This transition pattern is applicable when the existing application is written in a programming language (or languages) that is available on System z. Because System z supports most of the popular procedural and object-oriented languages including C, C++ and Java, this transition is applicable to many existing applications written in those languages.

When recompiling an application of any significant size, some "porting" work is usually needed. Although the language itself might be available on the source platform and System z, each platform's compiler or API set often differ slightly. These differences are small scale and will require some changes to the source, but usually they will not require the overall design to change.

7.3.4 TP4: Redeploy the existing application as it exists today

The focus of this transitional pattern is low *cost*. Your existing application is implemented in a platform-independent language, so it can, in principle, run on System z "as-is". The languages that fall into this category include Java and a number of interpreted languages such as Perl, JavaScript, PHP, and REXX.

The important attribute of applications implemented in these languages is that their executable form can be run on System z without changing or recompiling it. This does not mean, however, that the application will work correctly on System z; achieving that will usually require some changes.

Java is unusual in that it is a compiled language, but its executable form is not platform-native code. Instead, Java defines a platform-independent executable model that uses so-called Java byte-code instructions that are understood and executed by a Java Virtual Machine (JVM). Each physical platform has a JVM that can be used to run compiled Java code. The Java executable object is a Java class object or jar file, and its executable instruction set is called Java byte-code.

Beyond the Java language and its runtime JVM, there is a standard application middleware framework called Java Platform Enterprise Edition (JEE). The JEE provides a standard for a number of important aspects of enterprise application design, namely:

- ▶ Remote Procedure Calls (between JEE Enterprise Java Beans)
- ▶ Message-oriented transport (Java Message Service or JMS)
- ▶ Distributed transaction management (Java Transaction Service)
- ▶ SQL access to RDBMS (JDBC)
- ▶ Directory Services (JNDI)

Numerous commercial and open source products implement the JEE, including many that run on System z. Java JEE applications that run on a platform stand a good chance of being redeployed on a JEE environment that is available on System z without a major redesign or a significant amount of migration effort.

7.3.5 Summary of transitional patterns

Table 7-2 summarizes the transition patterns described.

Table 7-2 Summary of transition patterns

	TP1: Buy or re-code	TP2: Redeploy through conversion	TP3: Port by recompiling	TP4: Redeploy as is
Type	New application: either a software package or the application is written new	Tool-based conversion	Port or recompile	Redeploy
Focus	New function	Reuse	Minimize risk	Low cost

7.3.6 Mapping application patterns to transitional patterns

The application patterns discussed represent a set of common classes of business applications. Although not all applications will be a perfect fit to one of the three patterns described, most can be classified in that way.

The useful transition patterns described in 7.2, “Application patterns” on page 173 represent different approaches to transitioning applications to System z.

Table 7-3 relates each application pattern to the transition patterns and shows whether a certain transition approach is applicable to a certain application pattern.

Table 7-3 Mapping between application patterns and transitional patterns

	AP1: binary-compatible language	AP2: standard language	AP3: proprietary language
TP1: Buy or recode	When significant new function or redesign is needed.	When significant new function or redesign is needed.	When significant new function or redesign is needed.
TP2: Redeploy through conversion tools	Not applicable.	Not applicable.	When no new function or redesign is needed.
TP3: Port by recompiling. Portions might change	Not applicable.	Applicable.	Not applicable.
TP4: Redeploy as is	Applicable.	Not applicable.	Not applicable.

7.3.7 Middleware provisioning

The new application that is finally run on System z is likely to require one or more middleware products to support its design. The middleware needed for the new application will fall into one of the following categories:

- ▶ Transport or connectivity middleware
- ▶ Application framework middleware
- ▶ Database middleware

Each of these categories is discussed in more detail in the following sections.

Transport or connectivity middleware

A popular transport middleware product is WebSphere MQ. It provides a messaging-oriented connectivity solution between System z and most other platforms. WebSphere MQ provides a persistent, store-and-forward, messaging capability that is ideal for architectures that need assured message delivery.

For applications that emphasize performance and throughput over reliability, the WebSphere MQ Low-latency messaging product offers high-speed, low-latency messaging capability. Unlike WebSphere MQ, its emphasis is on these speed and throughput aspects, rather than assured delivery.

Application framework middleware

Application framework middleware provides entire *containers* for applications that allow the interface, connectivity, and data aspects of the problem to be decoupled from the application business logic.

These frameworks often also provide security and cryptographic services that can allow those functions to be decoupled from the application itself.

In the JEE space, WebSphere Application Server provides a standard JEE server environment for System z (and many other) platforms.

Database middleware

Assuming the old application used a database, then the target System z platform will need a new or existing DB2 installation.

DB2 will require sufficient DASD and processor resources to support the application's table spaces.

7.4 Switchover approaches

This section discusses techniques that you can use to plan the actual commissioning of the new System z-based application. The essential process is the redeployment of the production workload from the existing application to the newly transitioned System z application.

Now that you have determined the transitional pattern for your application, you need to plan for a switchover to the new application after it has been developed. The goal for the switchover is to minimize risk, because you must keep your business processes highly available at all times.

As mentioned in 7.1, “Introduction and overview of transitional approaches” on page 172, generally the method used to switch over from your existing application to the new one falls into these categories:

- ▶ The big bang approach, in which you shut down your existing application and start up the new one immediately after shutdown.
- ▶ The shadowing approach, in which both applications run simultaneously and duplicate or mirror data until you are satisfied that the new application can perform on its own.
- ▶ The phased approach, in which you transition only a portion at a time of the overall business application while other portions remain on the old platform.

Factors such as the overall challenge and risks determine how you execute the transition of your application. The following section describes switchover approaches and the challenges and risks for each approach.

7.4.1 The big bang switchover approach

The big bang approach is the simplest way to commission the new application. At the designated switchover time, the entire application workload is redirected away from the old application to the new one on System z.

The benefit of this approach is that the full workload is possibly transitioned to System z sooner than in other approaches. However, the switchover in production only takes place after proper testing and rehearsals.

During a big bang switchover, the old application is shut down (or idled) while the new application on System z picks up the full production workload. There are clearly significant risks with this type of transition that relate to its single point-in-time nature.

Figure 7-1 on page 180 illustrates this approach.

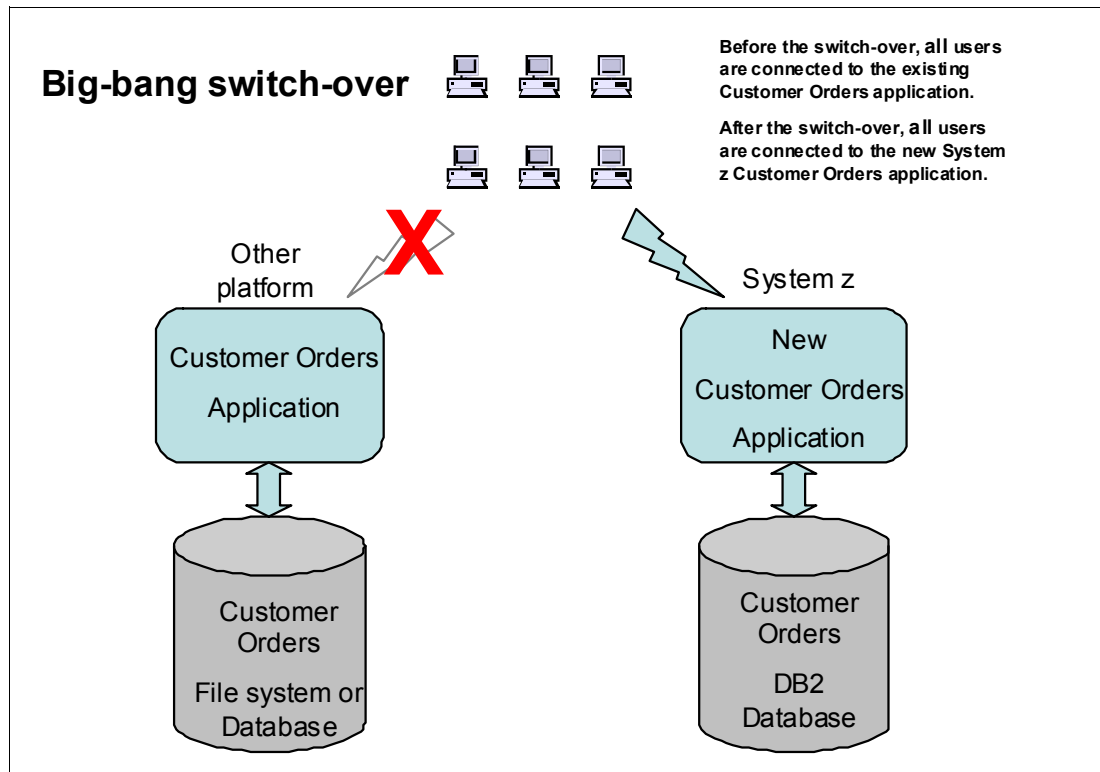


Figure 7-1 The big bang transition approach

Challenges and risks

This switchover approach entails the following risks:

- ▶ The entire user population or traffic workload is exposed to a switchover outage.
- ▶ A fallback to the old application will incur the same outage and risks.
- ▶ There is no built-in opportunity to “shakedown” the new application with a production workload.

Fallback

A decision to revert to the old application would necessarily involve a complete reversal of the switchover procedure. Depending on the nature of application system itself and the state of the old database relative to the new database, falling back might be difficult and entail a significant service outage.

7.4.2 Shadowing with replication

The shadowing with replication method is sometimes referred to as “parallel running”. This transition method supports the running of both the old and the new application at the same time, each using their own databases. Using this approach, the migrated users will be using the new application, while those not yet migrated are still using the old application.

Because both the old application and the new System z application are concurrently active, with this method some form of data replication is needed so that both applications see the same “view” of the available data.

Figure 7-2 illustrates this approach.

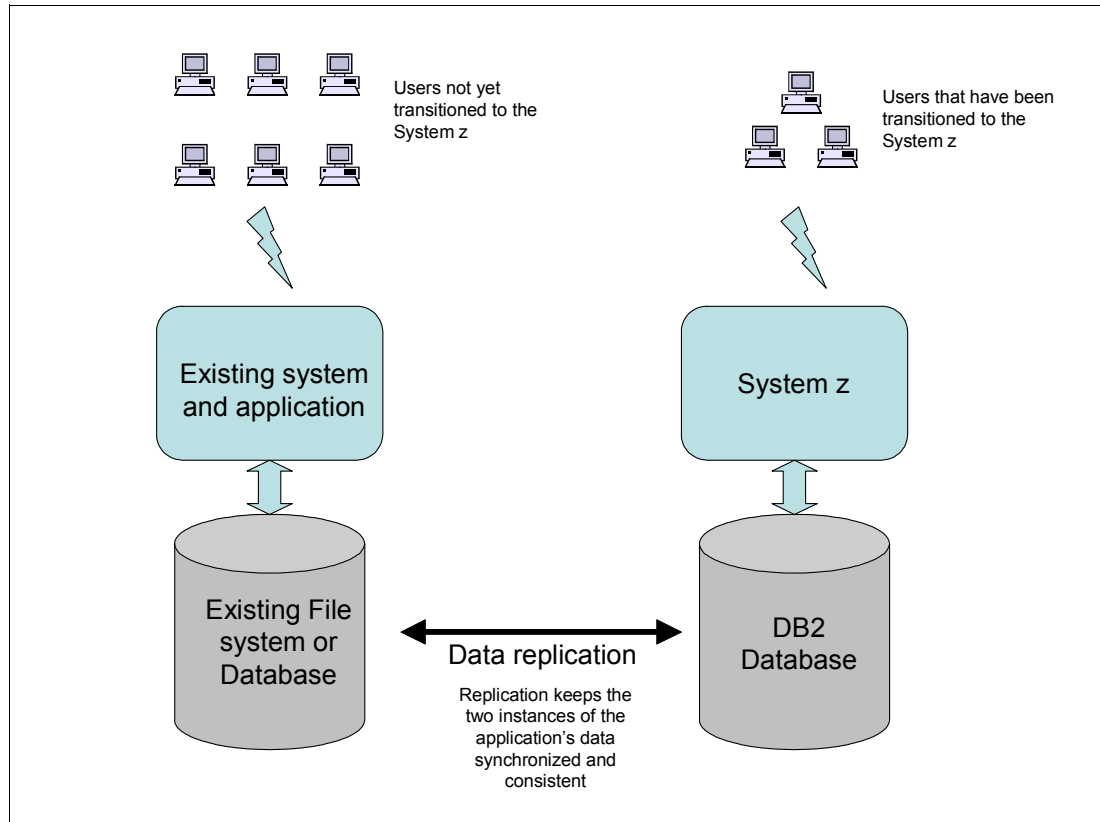


Figure 7-2 Shadowed or parallel transition with replication

Shadowing can have two forms:

- ▶ Your existing application and the new one can handle transactions simultaneously on separate databases.
- ▶ Or, one application handles transactions and periodically updates the database for the other application.

Challenges and risks

The shadowed transition approach is more complex than the big-bang approach for two reasons:

1. A method to either duplicate or divide the workload traffic must be devised.
2. A database replication method must be designed and implemented beforehand.

The workload duplication (fan-out) or division will require careful design to ensure that both the old and new databases are consistent and correct at all times.

Fallback

If the data replication method is working correctly, then the fallback procedure is relatively simple: redirect the entire user population or workload back to the old application system.

Depending on the design of the data replication feature, the new database can be kept aligned with the old database even after the fallback is implemented.

7.4.3 Phased switchover

A phased switch over approach attempts to divide the switch over into a number of smaller phases, each of which is easier to plan for and manage. The phasing can be applied in any of the following application dimensions:

- Phase by user organization or division

It is often advantageous to transition groups of users one at a time to the new application. Users can be grouped by organizational units (departments, offices, teams) or they can be grouped geographically. In either case, the transition plan will switchover one or more groups of users at a time.

- Phase by request traffic type

If the application processes a number of different types of requests, it may be possible to “phase” the switchover by request type. Each phase of the switchover would address some subset of total set of request types.

- Phase by application component

Many larger application suites are comprised of a number of components, or functions.

If these separate application components are sufficiently independent of each other, then it may be possible to switch over some of the applications components to System z before switching over others.

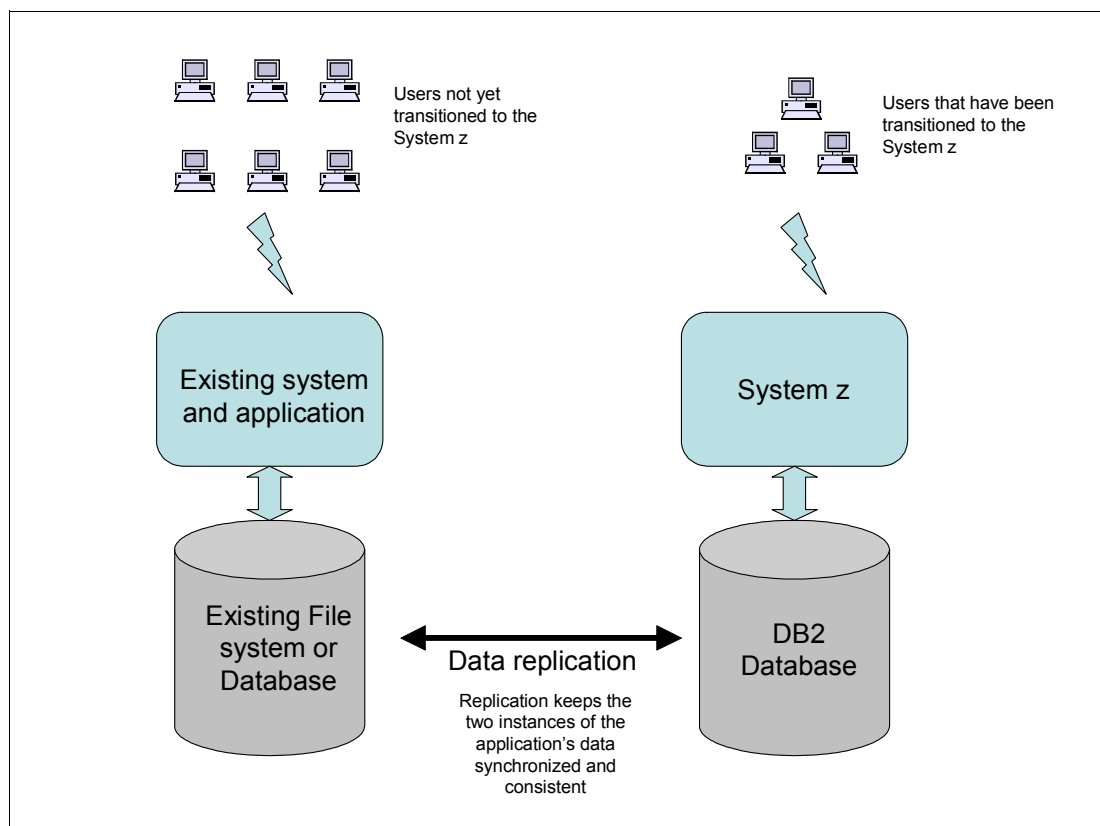


Figure 7-3 The phased transition approach

The phased transition approach can be thought of as a variation of either the big-bang or the shadow (with replication) approach. Using either those other approaches as a starting point, it is possible to apply phasing either to add manageability or lower risk.

Challenges and risks

The phased transition approach usually requires a more complex plan simply because there are more steps required to get to the final, complete transition. Further, a phased approach in the transition project will usually result in greater elapsed time and effort before a complete transition is achieved.

Fallback

If the data replication method is working correctly, then the fallback procedure is relatively simple: redirect the entire user population or workload back to the old application system.

Depending on the design of the data replication feature, the new database can be kept aligned with the old database even after the fallback is implemented.

7.5 Options for application modernization

The term “application modernization” embraces the desire to enhance or extend an older legacy application so that it can benefit from new design paradigms or new technology. Best practices for application architecture have evolved considerably over the last 20 years and some IT clients have seen opportunities to improve their older applications without replacing them.

Cost-effective modernization efforts require deep insights of your business applications, so that you can identify embedded business rules, restructure and remove dead code, and understand the impacts for code changes.

Modernization must address this issue and the complex dimensions of architectural challenges, including fragmented business processes, workflows, data, and tightly coupled application architectures.

Understanding your applications can improve the productivity of your IT staff and reduce maintenance costs through more automation and through eliminating the need to research, catalog, and assemble information for each service request individually.

There are many cost-effective approaches and techniques to transform your core systems into flexible applications and services. Finding the right software solutions to modernize your legacy systems is the first essential step.

7.5.1 The case for application modernization

Despite the challenges just enumerated, there are many reasons to transform and modernize existing applications:

- ▶ They run the business and contain critical business logic that is unique, difficult, and costly to replicate or replace.
- ▶ Modernizing existing applications allows you to reuse critical business assets and leverage an investment that you have already made.
- ▶ These systems are not necessarily broken, but they may be difficult or expensive to maintain, may not meet the present performance needs, and may not scale well enough.

There many tools that can assist with the effort, making it feasible to modernize these systems in a timely, cost-effective manner.

7.5.2 Aspects of application modernization

Application modernization can address different aspects of an application suite, from the way it interfaces with its users through to its internal architecture, as explained here:

- ▶ Modernize the user interface

Change a current text-based terminal user interface to one that gives the user a consolidated web look and feel, or add a custom *rich client* program.

- ▶ Modernize the application connectivity

Change the way that users or client applications can connect with the existing application, which can work with Java connectors and create Java wrappers. This can also involve exposing web service interfaces.

- ▶ Modernize the application architecture

Change the way that the application is constructed by making it more modular and making it easier to compose business services. These services represent mission-critical application components.

- ▶ Modernize the application code

This transformation involves the actual process of transforming the underlying application code to a new, modern language that supports deployment to any platform using whatever user interface (UI) is deemed necessary including Web, rich client platform (RCP), or text user interface (TUI), or traditional “green screen” for any type of processing (online or batch) or service creation.

7.5.3 Modernization using a service oriented architecture

There are different modernization strategies that meet the many disparate needs of enterprises. The scope of the modernization effort can be as narrow as upgrading the user interface, or as broad as rearchitecting the application as a service within a service oriented architecture (SOA) deployment.

SOA focuses on the concept of reusability, which means that application components must be built in a way that facilitates reuse and avoids creating components “from scratch” when existing assets can satisfy the requirements. The combination of existing data and applications on the System z offers a low-cost entry into the SOA model.

Enterprise transformation and application modernization are critical to an enterprise’s application development strategy, because the high cost and risk of rewriting existing applications is often an inhibitor. If a company can reuse existing well-tuned and proven code that has been implementing business function satisfactorily for years, then savings can be significant. Aside from the high cost and risk involved in migrating applications, performance, reliability, and scalability requirements often make the current environment the best choice.

When considering the reuse of existing assets, developers experienced in traditional languages must be considered an integral part of the overall business development team. As the availability of these development skills continues to tighten in the marketplace, it becomes more critical to create a development environment where existing assets can be maintained and extended using current development approaches and available skills. SOA has particular importance in the enterprise transformation practice, because it has the power to unlock existing applications and data and expose them as services, thus providing extended value to the business.

Considering the investments that companies have made in developing applications and the importance that those applications have to the core business, service enablement using

standard-based interfaces helps to extend the lifecycle of those applications and leverages existing investments.

The move toward SOA is not going to occur overnight. It is not an appliance to be installed on top of an existing infrastructure, thus making it SOA ready. The implementation process is a set of fine-grained changes applied to the existing architecture and the exploitation of new technologies that will gradually make the environment SOA ready.

The IBM enterprise transformation strategy supports this gradual, fine-grained process that will help to enable existing assets as services. It defines three solution frameworks, sometimes referred to as styles of transformation, to help clients convert IT assets from siloed applications to shared resources, and then to interdependent software components and services.

The strategic goal is to innovate by enabling the creation of new software components that have strategic business value and support an on-demand environment. However, that takes time, effort, and money, so, one immediate and tactical solutions are required as well.

For example, the ability to transform siloed applications to shared resources can enable better ways of interacting with clients, partners, and suppliers. Certain changes to applications will require immediate solutions that can drive business value, but they might not be as flexible in adapting to future changes in the business process. The three transformation styles of Improve, Adapt, and Innovate address both tactical and strategic service enablement solutions.

Improve, Adapt, and Innovate transformational styles

IBM identifies these three approaches for modernizing a client's enterprise. However, an IT organization will likely not use simply one style. Businesses often choose multiple styles of modernization for various types of solutions, so the deployment scenario is determined case by case. The transformation styles of Improve, Adapt, and Innovate cover both tactical and strategic service-enablement solutions. This section presents a more detailed look at these styles.

Improve

The Improve style is characterized by the use of new technologies to web-enable or service-enable applications at the user interface level without changing those applications and with minor changes or additions to the middleware infrastructure. Improve is often the first step toward SOA, where clients simply enable existing applications with SOAP or MQ protocols to facilitate integration. This style concentrates on transforming the user experience by providing a more sophisticated and productive user interface for applications.

Adapt

The Adapt style goes further in terms of application transformation and enhancement of the existing infrastructure. Adapt comes closer to full SOA architecture, but it requires more investment in business componentization and in IT services. The result is more flexibility for the business and the value that this flexibility represents.

Adapting existing connectivity enables broader application integration and provides the ability to incorporate core applications into more modern application flows. Adapting allows clients to leverage existing applications to develop better client, partner, and supplier relationships. But the connection of many existing applications to new applications and new architectures poses a compatibility problem: data formats, communication protocols, and existing programs frequently cannot communicate without an intermediary.

An Enterprise Service Bus (ESB) can act as this intermediary (that is, it can act as a broker) for data transformation, protocol transformation, and data routing.

The ESB is not a direct component of the Adapt transformation style, but it does serve as the hub to connect the applications, transactions, and services that are transformed using the adapt techniques. Other elements of the SOA reference architecture are also involved in this phase, including information and access services. The service management infrastructure is often planned and designed during this phase.

Innovate

The Innovate style of modernization is characterized by the creation of new applications that are fully compliant with the SOA model. Accordingly, with reference to SOA strategic approaches, this style is mostly top-down, where business processes are modeled using a modeling tool. Deployed applications invoke a service or services (applications) that make up the composite business applications or processes. This style can require that a totally new service or application be developed, or it can involve the transformation and reuse of an existing application to meet the business requirements. Transforming the application structure and architecture requires the highest degree of investment, but it pays off with the greatest business value and process flexibility.

In the Innovate style of transformation, core applications are restructured to provide the greatest amount of business benefit. This restructuring allows clients to more rapidly innovate and change their business processes using existing IT applications to create new and differentiated market solutions. To innovate, tools are required to design and deploy new applications, and a server is needed to orchestrate and direct the newly created business processes.

Another significant value that SOA brings to companies is an architecture that is programming language-neutral. This allows companies to continue to develop new applications using traditional languages, which leverages existing skills, tools, and investments. Being able to use traditional languages is important to System z clients because experienced application developers can continue to use their preferred languages and tools, providing improvements in productivity and economics.

7.6 Summary

This chapter discusses the planning and strategies needed to actually transition a workload to System z.

The concepts of application patterns and transition patterns are introduced. The chapter describes how these patterns can be used to guide the planning of the actual transition. Application patterns provide a method for classifying applications accordingly to difficulty and complexity, while transition patterns describe common scenarios for executing the transition.

Finally, the chapter illustrates options for modernizing the application either during the transition itself, or afterwards.



System z capacity planning, sizing, and TCO

This chapter discusses aspects of sizing and capacity planning on System z. The studies are based on industry experience, knowledge of machine architectures, current environment and performance metrics (if an application is being moved from distributed systems to IBM System z), or raw data specification (if an application is being sized “from scratch”).

Sizing involves a combination of “art and science” wherein practitioners use their skills and experience to convert real or estimated performance data from existing systems or specification data from non-existing applications to produce performance projections of how specific workloads will run when migrated to different platforms.

This is a relatively complex task, particularly when handling sizing across platforms built on top of different architectures. However, if the input data (hardware description, performance metrics, workload type) is known and trusted, there are sizing methodologies that can produce practical results which can then be used as a starting point to build a new solution hosted on the new platform architecture.

If you are unfamiliar with the required techniques, help is available through the IBM Global Techline or through your local IBM technical resource. These contacts can advise you how to gather the information that is required for a proper sizing or capacity planning analysis.

IBM Global Techline IT professionals provide presales technical support including IT Optimization analysis through sizing and capacity planning studies, thereby enabling solutions and projects. Techline provides the resources needed to analyze each situation on a case-by-case basis and provide you with a consolidation sizing report, a total cost of ownership (TCO) comparison, and the full configuration for your proposal.

Be aware, however, that sizing results are simply guidelines and starting points. It is the responsibility of the local IBM team to ensure the sizing has been performed by taking into consideration all of the client's specific needs, and to accommodate the local business requirements and mitigate all identified risks.

The purpose of this chapter is to help you understand the importance and complexity of sizing, capacity planning, and TCO studies. Because the quality of the output depends

directly on the quality of the input, this chapter also has the objective of identifying the characteristics of a quality input. The ultimate goal is to enable you to provide quality input to the practitioners and to receive back a quality report.

Note the following helpful resources:

- ▶ IBM Redpaper™ *Linux on System z, an end-to-end view*
[http://www-03.ibm.com/support/techdocs/atsmastr.nsf/84279f6ed9ffde6f86256ccf00653ad3/1c76a8c5aa63c3b28625723a0026816b/\\$FILE/Linux on System z, an end to end view.pdf](http://www-03.ibm.com/support/techdocs/atsmastr.nsf/84279f6ed9ffde6f86256ccf00653ad3/1c76a8c5aa63c3b28625723a0026816b/$FILE/Linux%20on%20System%20z,%20an%20end%20to%20end%20view.pdf)
- ▶ IBM Redbooks publication *OS/390 MVS Parallel Sysplex Capacity Planning*, SG24-4680
<http://www.redbooks.ibm.com/abstracts/sg244680.html>
- ▶ IBM Redbooks publication *IBM zEnterprise System Technical Guide*, SG24-7833
<http://www.redbooks.ibm.com/abstracts/sg247833.html>
- ▶ IBM Global Techline Center of Excellence, to be engaged through your local IBM representative

8.1 Overview of sizing, capacity planning, and TCO considerations

Because of the many benefits offered by System z (including low floor space requirements, unmatched virtualization capabilities, energy efficiency, security, and reliability), mission-critical enterprise Linux applications are increasingly being deployed to IBM System z servers, which are arguably the most reliable and available servers on the market.

As in any IT environment, elements such as sizing, capacity planning, and total cost of ownership (TCO) have always been important factors in supporting a decision to run or consolidate new applications onto a given platform. These same elements are essential in minimizing the possibility of critical situations, and they are key to enabling IBM to earn from its clients the high levels of satisfaction we strive to deliver.

Part of the advice given in this chapter specifically targets System z, which is a flexible platform that offers alternatives in terms of hardware and software. For instance, z/VM hosted Linux images running on System z are an efficient and cost-effective alternative to distributed servers. The traditional z/OS platform offers the unmatched capabilities that helped to build the System z reputation as the most reliable server in the industry. It offers clustering capabilities such as Parallel Sysplex technology that provides the highest availability levels. Throughout this chapter, we consider these differences and examine how to optimally use each configuration.

8.2 The importance of sizing

Proper sizing is considered the starting point for any relocation or consolidation project. You need to understand the requirements to move workloads from one platform to another. In this chapter, we will always consider the target platform to be System z. Therefore, the relocation or sizing discussion will always cover workloads being hosted on “non-z” platforms.

For our purposes, *sizing* is defined as the exercise in which we use an IBM-developed methodology to best quantify the requirements of a given workload to be migrated from distributed servers and properly run on System z in the most efficient manner.

Sizing is not an exact science or a magical process. It depends on the quality of data collected; as the saying goes, “garbage in means garbage out.” Sizing is a technical component that presents a technical solution to be integrated into the business solution.

Proper sizing requires the best and most accurate data and description of the current environment. Although using tools such as Right Fitting Applications into Consolidated Environments (RACEv) allows sizing to be performed without real utilization data by using statistics to model the environment, at the time of writing, IBM continues to work on validating the statistical model against real data.

A useful description of the servers being consolidated includes make and model, number of chips and cores, clock frequency, memory utilization, processor utilization, and a description of each application running in each server.

Note: The number of intervals, duration of each interval, dates, and time stamps should be common to all servers being consolidated.

The data needs to be representative of your system, and needs to be based upon an accurate picture of where you stand today. This means the data collection has to be recent. Use care when selecting the “data/time” granularity, which is the amount of data collected in a given time interval, because employing long intervals can lead to averages that may conceal the true performance of your system. Also, the workloads being measured need to be representative of the real production environment.

In a nutshell, sizing is the “translation process” from one platform to another. It is the science and the art that allow an “apples to apples” comparison across servers from different architectures. However, you can also encounter a situation where sizing is done for a totally new non-existing application, and which is based on parameters such as expected transactions per second, database size in GBs, and so on.

This chapter will help you to gather the required data, point you to the existing IBM resources to support you in this process, and explain the importance of how and why you are collecting the various pieces of information.

First, you need to consider what and whether current performance data is available for your servers. If regular reports are being produced to track service level agreements (SLAs), there will usually be a specific tool employed to process these. If you do not have any tools, we make recommendations of how input data can be gathered using freely available tools or by using system level commands.

Proper and reliable sizing is essential. A significant underestimation will cause problems when the new system enters the production phase. A significant overestimation can result in resource optimization targets being missed.

One important factor to mention is that the sizing process, although required, often is not all you need to effectively prove the value of the System z platform. A Total Cost of Ownership (TCO) is an effective mechanism to demonstrate the value of the z platform. It elevates the conversation from the technical aspects to the financial vocabulary, which is often better understood by the Cxx executives such as CIOs and CFOs.

IBM offers more than one path to establish System z cost and value. The following list contains a few examples of System z TCO resources:

- ▶ RACEv and RACEzOS
- ▶ EAGLE
- ▶ SCORPION
- ▶ ALINEAN

8.3 Capacity planning importance

Capacity planning is the study that verifies and analyzes the hardware requirements needed to accommodate the growth of workloads that are already running on or sized for a System z machine. Capacity planning takes into consideration the performance of the existing System z and the growth projected for the existing workloads. After the analysis is completed it will produce a recommendation for the System z hardware to absorb the growth within the platform. It takes into account the growth of today’s workloads that already run on IBM System z to enable them to be run in tomorrow’s IBM System z.

For example, assume a client already has an IBM System z server, and has seen an increase in the current workload’s usage. The current machine does not have enough capacity to process the inflated workload within the agreed SLAs. There is a need to add more hardware to bring the SLAs to an acceptable level. The questions become: how many additional

processors are needed? And should the client replace the existing machine with a newly announced System z server?

Another scenario is described when a client wants to add a new workload to the existing System z server. How much extra processor consumption can be expected?

Capacity planning provides the method and tools to handle such situations, and it helps you to understand and define the requirements to ensure adequate processing.

Ultimately, capacity planning helps you to verify and analyze your current system configuration and performance, and then projects the estimated future environment that will be required to cope with the anticipated growth of the workload. It is important because it considers the future: the entire system will grow over time, and the idea is to devise a system that anticipates and supports that growth.

It is also important to correlate the future needs of a system with *any* planned changes in the underlying business areas. For example, there are times when a contraction will be required. When capacity planning is well executed, it will provide useful system performance and cost management.

Review your capacity planning at planned intervals that relate to the rate of change seen on the given system. Otherwise, there is a risk that workloads growing faster than predicted remain undetected, which can cause sudden contention for resources and even downtime.

A critical element of the capacity process is the verification of how close real performance is to projected performance. Clients can sometimes add workloads that were not part of the capacity study, or their actual growth rates can be larger than the rates used in the study. This emphasizes the importance of monitoring the system on a regular basis.

Whenever possible, verify actual results against projected results; that is, undertake the classic “before and after” comparison. The goal is to show that the base capacity planning (when from one System z server to another System z server) or the sizing projections (when consolidating) are allowing the system to run with the expected levels of performance.

Tuning can also be part of the process, by optimizing the system to run as efficiently as possible. If tuning is performed it is best to make only one change at a time, and then examine the effects and verify that change, before moving on to make other changes. Unravelling the interdependent effects of multiple changes can be difficult and sometimes impossible.

The goals of understanding the before and after situations, whether obtained through capacity planning (System z to System z) or through sizing (non-System z to System z), are to better understand the results, validate the input, better forecast future growth, and create a larger database of real cases to be fed back to development. This process enables each of us to participate in the enhancement of existing tools and methodologies.

8.4 Data collection and analysis

Data collection is an important step in any sizing effort. It involves gathering the data elements required to enable further analysis. As a starting point, the collected data needs to be adequately representative of the current system. Knowledge of how the system performs is useful, such as when peak loading periods occur and how they relate to any SLAs in place.

The most reliable way to perform capacity planning or sizing is to use real data as input. Capacity planning input can be SMF records (in the case of a z/OS environment) or z/VM monitor data (in the case of a z/VM environment). Note that for a Linux guest, the z/VM

monitor data can be used to analyze the resources used by the guest. Sizing input requires, at a minimum, the processor family (x86, RISC), number of chips, numbers of cores, clock frequency, and workload characteristics. Processor utilization is desirable and required to enable the use of the best techniques available, although the lack of this information can be replaced by statistical analysis. However, nothing is as useful as using actual data.

Data analysis is the final step in the capacity planning or sizing effort. In the data analysis process, the data previously obtained is analyzed using the tools and techniques available. The final deliverable is a report projecting the utilization and defining the requirements on the new environment.

8.4.1 Data collection

Data collection is the process of gathering server resources information, such as processor consumption, memory usage, and so on. When there are two or more servers being studied, the data collection needs to be performed in as synchronized a manner as possible, or “quasi-simultaneously,” meaning that you need to collect data from all servers being consolidated during the same time intervals and using the same time-stamp format. The goal is to allow you to stack data from all the servers into a single stacked graph, where the overall peak represents the peak of the entire interval for all servers combined. Note that the sum of the peaks is usually larger than the peak of the sum. For the most efficient plan, the goal is to use the sum of the peaks. Data collection must be done within a period that represents the typical business processes represented by the client’s workloads. Invariably, this will involve consideration of the peak loading periods. If you configure a system to perform adequately during expected peaks, the system should be able to perform well on all workloads being consolidated. The periods chosen should ideally show the most important high level resources, such as processor, memory, I/O, and network, being exercised at the typical levels seen when the system is busiest.

A more aggressive technique is available that uses the 90th percentile instead of peaks. The 90th percentile value for a set of values states that at least ninety percent (90%) of the values in the set are less than or equal to this value. The top 10% of the available samples are then disregarded.

Data collection for sizing is conducted with the aid of appropriate tools. Very often these products will be present on a system and already well known to capacity planners. For environments with UNIX or Linux that do not have such tools in place, the NMon tool is useful. This tool has versions for both Linux and UNIX. The NMon tool allows you to collect various pieces of information about current behavior of the servers, including the following examples:

- ▶ Processor utilization
- ▶ Memory use
- ▶ Kernel statistics and run queue information
- ▶ Disks I/O rates, transfers, and read/write ratios
- ▶ Free space on file systems
- ▶ Disk adapters
- ▶ Network I/O rates, transfers, and read/write ratios
- ▶ Paging space and paging rates
- ▶ Processor and operating system specification
- ▶ Top processors
- ▶ IBM HTTP web cache
- ▶ User-defined disk groups
- ▶ Machine details and resources
- ▶ Network File System (NFS)

Although often not as powerful or easy to use, operating systems offer their own standard monitoring tools. For instance, operating systems such as UNIX or LINUX offer the System Activity Reporter (SAR¹) command, which can also collect data on system activity.

An alternative is to use Virtual Memory Statistics (VMSTAT²). VMSTAT is a computer system monitoring tool that collects and displays summary information about operating system memory, processes, interrupts, paging and block I/O, traps, and processor activity. Users of VMSTAT can specify a sampling interval that permits observing system activity in near-real time.

The VMSTAT tool is available on most UNIX and UNIX-like operating systems, such as FreeBSD, Linux, or Solaris. The syntax and output of VMSTAT often differ slightly between different operating systems.

Most recently, the IBM Advanced Technical Skills (ATS) organization offered a method to collect performance data. The tool is named the IBM ATS Server Consolidation Monitor (ATS SCON Monitor). This tool can be used to conduct server consolidation studies. It gathers architecture and performance data from Windows computers and UNIX and Linux hosts.

8.4.2 Workload characterization

After the data is collected, it must be analyzed. A main aspect to be considered is the general characteristics of the application.

In the authors' view, the zEnterprise, or z196 for short, is a game changer. With the fastest production processor clock ever produced, at 5.2 GHz, the silicon behind the z196 is still a piece of art, almost one year after its release. This new capability presents the z196 as a possible candidate to host workloads that were not recommended when the target systems were still IBM System z9® or z10 servers. High processor usage is not the best fit to be run at the System z (zEnterprise, in this case) every time, but it needs to be considered and evaluated on a case-by-case basis.

IBM System z workload migration characterization

This section includes examples of workload characteristics that are frequently encountered. How these workloads are handled when running in a System z environment will depend on which workload type you are considering. When consolidating distributed systems onto z architecture, you normally can achieve better consolidation ratios with workloads that are I/O-bound. This is due to the unique characteristic of the mainframe architecture which presents a dedicated I/O subsystem. The I/O subsystem offloads the I/O processing from the main processors, releasing them to perform real work. In a distributed world, the main processors are also responsible in executing I/O operations.

Characterization of the workload type is important because it is key to defining what is called the “workload factor”. The workload factor is a constant, part of a mathematical calculation, that enables us to convert the distributed systems' capacity to MIPS³. This is required to enable comparisons between IBM System z and the distributed systems, thus establishing a single reference.

Because the workload factor alone can be responsible for results varying from - 200% to +200% (a 4x window), its proper definition is crucial during the sizing process.

¹ <http://linux.die.net/man/1/sar>

² <http://linux.die.net/man/8/vmstat>

³ MIPS= Millions of Instructions per Second

The application characterization influences the consolidation ratio for the analyzed workload. Usually, applications that are I/O-intensive have a higher consolidation ratio than applications that are processor-intensive.

Mainframes are available in scalable configurations, from entry-level machines offering equivalent capacity of medium to high-end distributed servers, to the largest mainframes capable of consolidating many distributed server cores. Machines towards the fully configured end of the scale tend to be best utilized when their resources can be placed in shared pools, which can be utilized by a large number of subsystems and workloads. Take care to select workloads for consolidation that benefit from being in such a shared resource environment. Fortunately, this applies to the large majority of typical, real world workloads.

It is still possible, however, to run workloads that do not share resources well under certain circumstances, for example a highly processor-intensive application that relies heavily on numerical calculations and then occasionally writes results to a disk. These workloads can run successfully as LPAR partitions with dedicated processors, or in the case of the new IBM zEnterprise, as dedicated POWER7 or x86 blade environments. Prior to the introduction of zEnterprise, the mainframe was not well equipped to run these workloads but today, a z196 server itself has much faster processing. However, such workloads can be more efficiently run on zBX blades attached to the z196.

Figure 8-1 on page 195 illustrates the types of workloads that are well-suited or best-suited for consolidation. The message in this figure has changed since the introduction of the IBM zEnterprise. Prior to the zEnterprise system, when System z did not provide POWER7 and x86 support by means of the zBX blade extension, the most suitable workloads for consolidation were I/O-bound workloads and the least suitable were processor-bound workloads.

With zEnterprise, processor-bound workloads can be equally suitable candidates for consolidation as I/O-bound workloads. Generally speaking, an I/O-bound workload fits better on a z196 server and a processor-bound workload fits better on a blade extension (either POWER7 or x86). Keep in mind, however, that this is a generic statement and might not be applicable in your case. Perform a “Fit for Purpose” study to analyze where certain workloads will fit best on a zEnterprise.

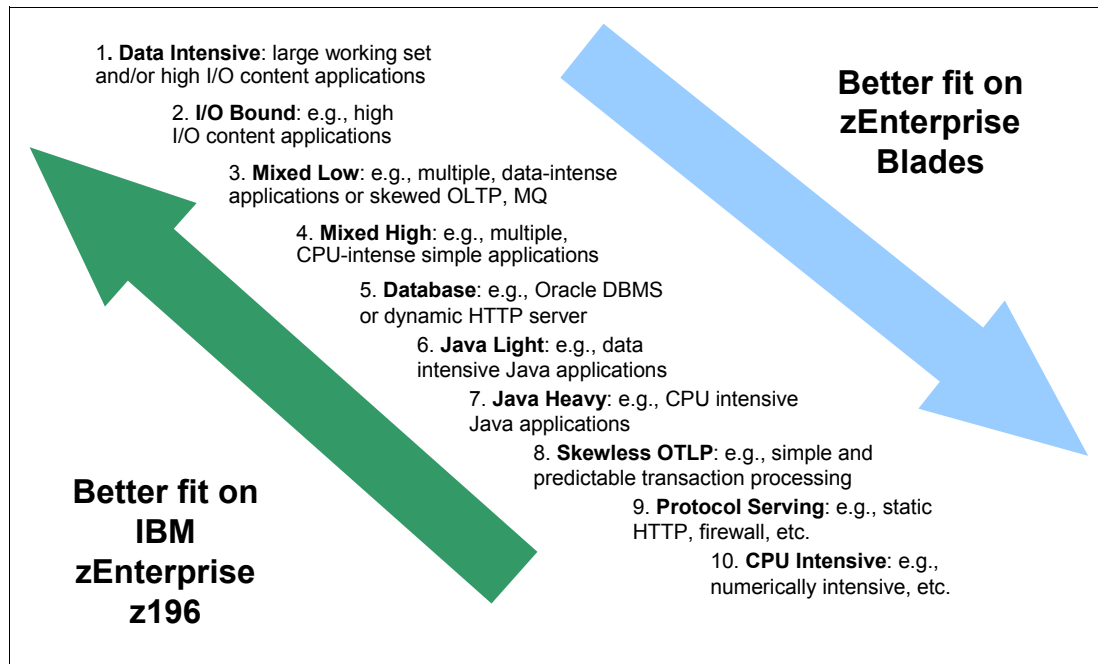


Figure 8-1 Mainframe workload placement characterization

The following is a brief description of each of these workloads:

1. Data intensive - A large working set or high I/O content applications, for example data warehouse applications.
2. I/O bound - High I/O content applications, for example database servers with high I/O transactions.
3. Mixed low - Applications with OLTP characteristics and high I/O transactions, for example WebSphere MQ servers.
4. Mixed high - Applications with OLTP characteristics and more processor activity, for example multiple simple applications with processor-intensive characteristics.
5. Database - Database OLTP, with processing spent on database access transactions, for example Oracle database servers with a high volume of PL/SQL transactions (resolved with database resources).
6. Java light - Java applications with a high volume of I/O-intensive transactions.
7. Java heavy - Java applications with a high volume of processor-intensive transactions.
8. Skewless OLTP - OLTP applications with simple transactions processing, for example mail servers with anti-virus checking and DNS servers.
9. Protocol serving - Static protocol servers, for example HTTP servers, firewalls and proxy servers.
10. CPU intensive - Processor-intensive servers, for example servers for numerical simulation.

Note: When workload characterization is not carried out properly, the sizing end result may not reflect the actual workload analyzed, producing less than desirable results (under-sizing or over-sizing).

For example, if you have an application with a high intensity of network traffic or heavy use of the disk storage subsystem, it will be characterized as type “I/O intensive”. This means that

when it runs on System z, it will make extensive use of the I/O subsystem, which is a known strength of System z. Another benefit is that the processor resource consumption, including any applicable specialty processors such as IFLs, CPs, zIIPs and zAAPs, is minimized.

Other areas that are also important in data analysis include current transactions per second (TPS), current response time, expected response time, elapsed time per batch job, agreed service level agreement, processor analysis, network analysis, memory analysis, disk analysis, and so on.

In addition to the characteristics of the applications, the following aspects are important in data analysis:

- ▶ Number of concurrent transactions per second
- ▶ Current response time
- ▶ In-host response times
- ▶ Elapsed time per job in batch
- ▶ Any agreed service level agreements in place that relate to capacity and delivered performance
- ▶ processor consumption analysis
- ▶ Network usage analysis
- ▶ Memory usage analysis
- ▶ Disk utilization analysis

Each aspect is explained in more detail in the following sections.

Current transactions per second

You must bear in mind the concurrent number of transactions per second running in the application environments. The number of transactions is usually measured in the application itself or with the help of specific tools. It is common in database systems, as the system itself provides the average and peak number of transactions per second. It is the same for application servers, mail servers, web servers, and others.

It should be assumed after migrating the application to IBM System z that at least the current peak number of TPS will be executed. Getting these figures may involve running stress tests. In various cases, the real number of TPS cannot be reached in simulated environments without a capable load generator. It is then necessary to measure the system based on consumption of resources that can be generated, and then extrapolate the figures observed until they meet the required TPS level.

Current response time

One empirical way to determine an application's response time is to obtain feedback from the application's users. Users are often not reticent about providing their views. Their level of satisfaction with a given application may be seen as the response time "thermometer", and indicate whether or not the levels are acceptable. A benchmark script based on typical workload elements is run on several systems, and the results are compared with those found acceptable over time. Benchmark or target figures suggested by the software vendors or related user groups can be useful values to aim for.

Note: Response time as perceived by users can be subjective, so it is important to determine whether network time and time spent on the user interface is or is not part of the response time measured. In a world filled with diverse user interface devices, end-to-end response times can vary greatly.

Perform a thorough analysis to ensure high quality data is collected. Sensible guidelines usually define response time in three different categories:

- < 0.1 seconds** Users perceive a near-instantaneous response time from the system, which is usually noticed and appreciated.
 - < 1.0 second** Users perceive a relatively rapid response from the system. Most users will be satisfied with this response if it generally remains in the subsecond band.
 - 1.0 - 2.0 seconds** Users perceive there is delay in the response from the system, but it is not experienced as disturbing.
 - 2.0 or more seconds** Users perceive there is a considerable delay in the response from the system. If this is consistently much longer than the normally expected response time, urgent action may be required, and checks in the network environment, processor consumption, memory usage, and I/O responsiveness should be carried out. Take action to reduce the response time to lower levels.
- Note that depending on the application type and the particular workload being executed, however, a long response time for the task may be expected for certain transactions.

The response time of the application to be moved is measured before starting the migration itself, because it provides a useful reference point. In this case we recommend the adoption of a methodology that allows measurement of the average response time of the main application types that run on your system, as shown in Table 8-1.

Table 8-1 Sample of application ID and current response time

Application ID	Current mean response time
App_A_0001	1.0 ms
App_C_0005	13 ms
App_A_0009	1.0 s
App_B_1002	15 s
App_C_0045	15 ms
App_B_1001	29 s
App_A_1004	20 s
App_C_1092	18 s
App_B_0001	2.0 ms

The response time is measured with the help of specific tools depending on the type of workload and their components. For instance, database management systems (DBMS) or application servers offer the tools to enable such metrics.

Expected response time

The expected response time target should then be based on the standard response time previously measured in the original environment. For example, using the response times measured in Table 8-1 on page 197, we created a table with the expected response times, as shown in Table 8-2.

Table 8-2 Sample of application ID and expected response time

Application_ID	Current response time	Expected response time
App_A_0001	1.0 ms	same
App_C_0005	13 ms	10 ms
App_A_0009	1.0 s	same
App_B_1002	15 s	7 s
App_C_0045	15 ms	8 ms
App_B_1001	29 s	10 s
App_A_1004	20 s	7 s
App_C_1092	18 s	10 s
App_B_0001	2.0 ms	same

On the new system, response times should be at least equal to the original system response times, but it is desirable to have an improvement in response times after the migration to the new system.

In migrating from an older system to a modern and faster one, users usually expect an improvement in performance and response time. Along with the System z architecture comes a variety of mainframe tools that are dedicated to providing a larger degree of visibility into system resources. These tools are discussed in detail in the following sections.

- ▶ “Processor analysis” on page 200
- ▶ “Memory analysis” on page 207
- ▶ “Disk analysis (I/O)” on page 208
- ▶ “Network analysis” on page 210

Batch - elapsed time per job

For companies across all industries, batch processing is still a fundamental, mission-critical component. Examples of batch processes commonly used in industries today include:

- ▶ Generating reports of all daily processed data
- ▶ Printing or sending bi-weekly account statements to all clients of a bank
- ▶ Paying salaries for all employees
- ▶ Running analytics on a large data warehouse (although real-time analytics is picking up)
- ▶ Archiving historical data at the end of each month
- ▶ Optimizing databases
- ▶ Creating backups of files and databases for disaster recovery (DR) purposes
- ▶ Processing files with large amounts of data from business partners

Batch processing provides significant benefits when there is repetitive logic or execution involved, for example when static data is read only one time, cached, and then reused throughout the program. Another example is when there is an update (SQL query) in a database status field table, all 10000 rows. Batch processing in this situation is far more efficient than 10000 online transaction processing (OLTP) programs, each running a query to update the status field, one row at a time.

For this purpose, data analysis for batch processing in a migration project can be important. After migration, the new system should have more batch processing power than the old

system. You must be cautious, though, and carefully observe the critical aspects for each batch job.

We suggest that a batch window comparison should be made for each batch job. Using this method you can analyze the effects on the batch jobs when running on the newer and more powerful processor.

Note: When migrating from other systems to IBM System z, you might need to convert to another batch processing programming language that is supported on IBM System z. See 4.10, “Programming environment” on page 107 for more information about this topic.

Normally for most companies, the batch job is submitted during a specific time window, called the *batch window*, which typically starts after business hours to avoid interference with OLTP. We consider this time profile to be an important aspect in the analysis of data related to batch, because you must avoid any interference between BATCH and OLTP workloads. As a rule of thumb, OLTP workloads run in the daytime prime shift.

Tools such as Tivoli Performance Modeler (zTPM) or Batch Workload Analysis (BWA) are often used to estimate the batch window and to project its duration during the analysis phase.

Figure 8-2 on page 199 shows, over a period of 24 hours, the normal behavior for BATCH and OLTP workloads.

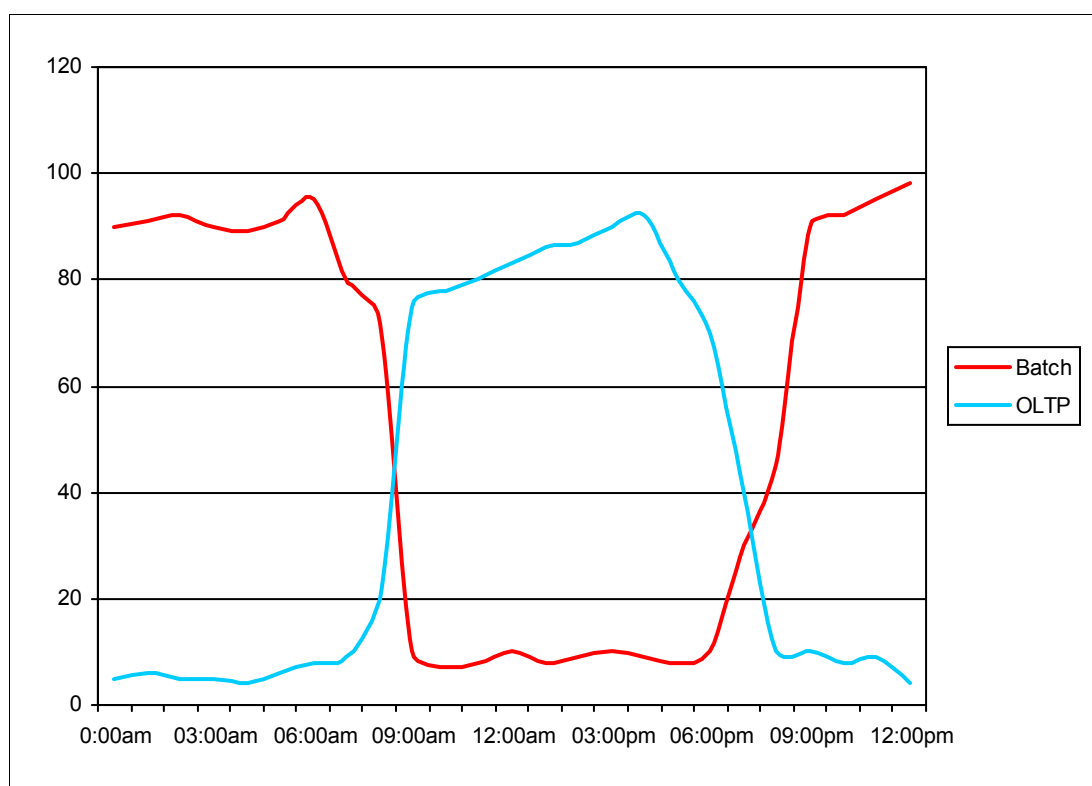


Figure 8-2 Sample of batch and OLTP workload on typical day

Service level agreement (SLA)

A service level agreement (SLA) is a negotiated agreement between two parties where one is the client requiring the service and the other is the service provider. This can be a legally binding formal or informal “contract”.

It is common for an SLA to specify the acceptable levels of parameters such as availability, serviceability, performance, and operations, or other attributes of service such as billing. The SLA can be used to include specifications such as those listed here:

Availability	For instance, during one year, the system cannot be unavailable for more than 5.3 minutes (99.999% of availability). Availability may also detail planned and unplanned outage tolerances.
Incidents	For instance, during one year, the system cannot be subject to more than 01 (one) error per every 1,000,000 processed operations.
Performance	For instance, the processing of all payroll operations should not take more than 0.2 seconds per employee, or 2 seconds for each 10 employees.
Priorities	For instance, concerns are dealt with according to their priority: “Urgent” concerns should be resolved in less than 8 hours, “Priority” concerns should be resolved within the first 24 hours, and “Regular” concerns will have up to 72 hours to be resolved.

The goal for migrating to any new platform should be that it matches or is better than existing SLAs.

Meeting the SLA is a predominant factor to be considered in any sizing. The entire sizing result will be validated by whether or not the SLA is met. Keep in mind that after an institution in any industry has agreed an SLA with its clients, it needs mechanisms to continue to meet this SLA. Therefore, we must consider SLA, regulating OLTP (“Current response time”) or BATCH (“Batch - elapsed time per job”) workloads as major factors in achieving the sizing objectives.

SLA for OLTP workloads

The main SLA factor involved in handling OLTP workloads is application response time.

SLA for batch workloads

There are two major factors involved in any SLA for batch workloads: the elapsed time and the completion time. With the elapsed time, you can see how long it took each batch job to run. The completion time enables you to verify whether the job required additional time outside of a predefined batch window.

8.4.3 Processor analysis

One of the most important elements to be factored into a sizing is the analysis of processor utilization. Processor utilization can be considered in the analysis in three ways: estimation (best guess), statistical estimation (better), or real measured utilization (best). The real measured utilization option tends to provide the best input because it eliminates the guesswork. As good as any mathematical model, the statistical analysis is a model or approximation. That is the reason why nothing replaces actual, measured data.

When you use real data previously gathered using one of the many tools described earlier, avoid mixing data from different dates and different time slots when consolidating the servers into a visual form (the processor utilization stacked graph). If you do not use a common base to create the stacked graph, the peaks and valleys visualized may not represent what is actually happening, which will compromise the credibility of the sizing. Therefore, always collect information about resource usage in exactly the same time frame and on the same day. See 8.4.1, “Data collection” on page 192, for more information about this topic.

When consolidating two or more servers, always perform data collection as simultaneously as possible on each server so that you can stack the workloads on a chart and accurately consider the coincidence of peaks over time.

As an example, Figure 8-3 shows the consumption of processing resources of multiple servers running an intranet webserver for a fictitious client.

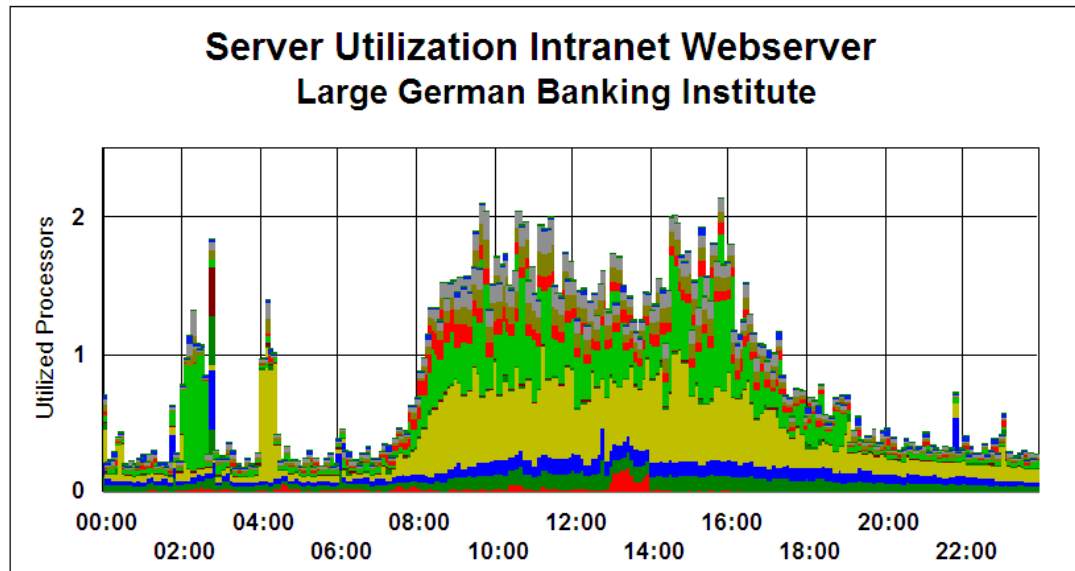


Figure 8-3 Sample of processor consumption for a fictitious client

Notice the coincident peak processing around 9:30 to 11:30, and around 16:00. These peaks are the accumulated peaks for all servers, representing the peaks of the sum and not the sum of the peaks. The sum of the peaks requires more resources than the peak of the sum. The peak of the sum is absorbed on the System z due to its excellent virtualization capabilities. This example illustrates why the samples need to be synchronized in terms of date, time stamps, and the duration of each interval representing the samples. The chart shows both the need for, and benefits of, a synchronized data collection time across all servers.

Average processor consumption

Average processor consumption must be carefully evaluated. It is a key value, so take care to obtain a representative figure from the analysis. Sometimes it is required to exclude data from holidays and weekends to remove abnormal low utilization samples. Other times the samples that need to be disregarded are related to spurious peaks that do not represent the normal work being executed in the servers. Analyze each case to derive the best average processor utilization number that you can from the existing data. For example, in Figure 8-4, the average processor usage required by SERVER01 is represented by 14.9 physical processors.

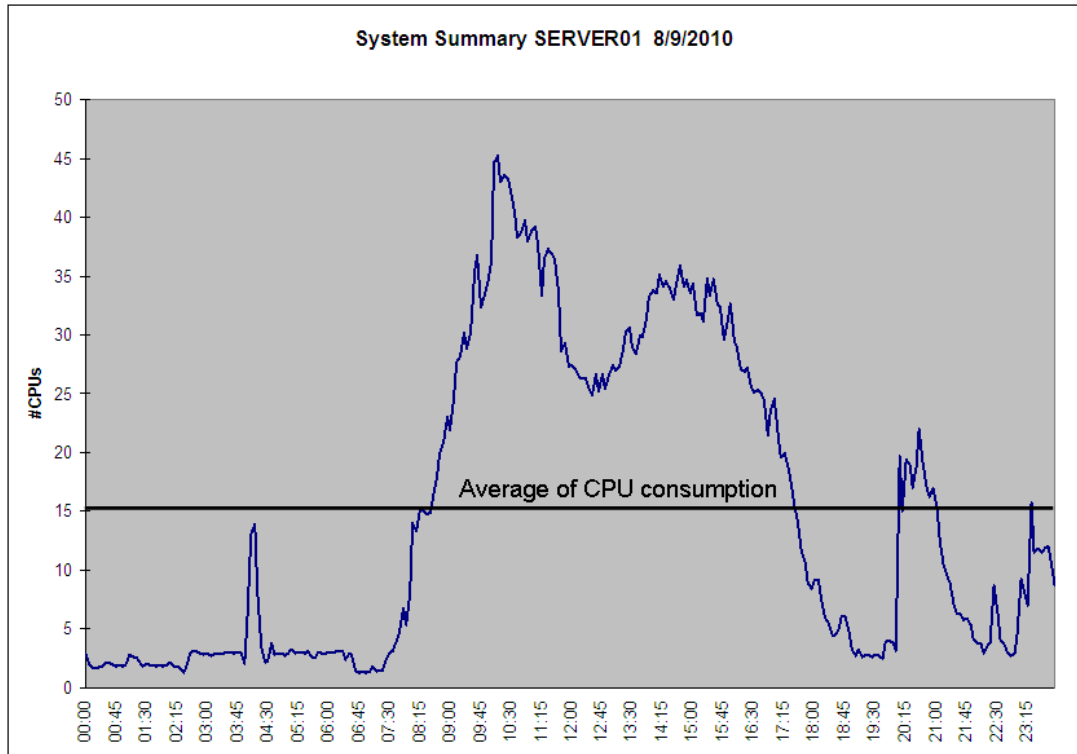


Figure 8-4 Sample of average processor consumption

Be aware that assuming that the processor average represents the entire interval does not produce accurate results and the target server will be underestimated, given that the resources allocated will not be capable of absorbing the workload during heavier processor usage. The average processor consumption fails to account for the real processor consumption during the most important processing period of the day, which is between 8:00 and 18:00.

One way to make the use of average processor consumption more meaningful is to target only the interval where there is heavy activity. Assume we “zero” into the interval represented from 8:00 to 18:00. Figure 8-5 on page 203 represents this new scenario.

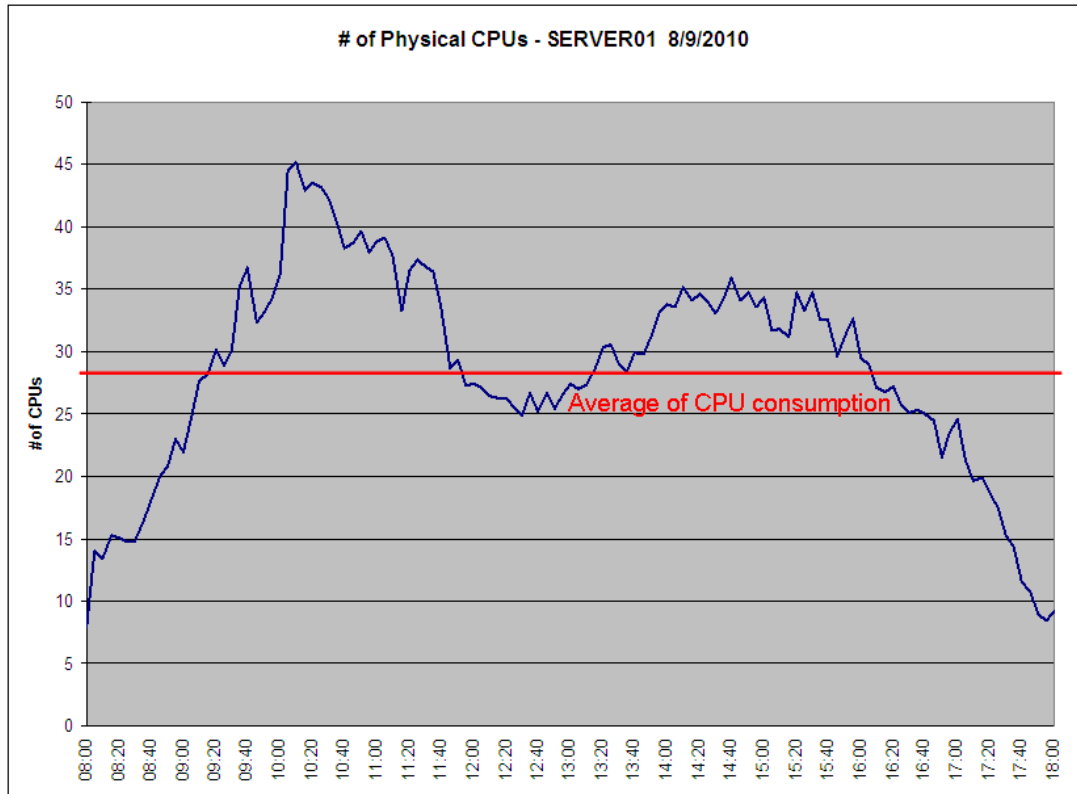


Figure 8-5 Changed interval of analysis

The situation has changed dramatically and we now have an average consumption of about 29 processors, which is much closer to the peak processor utilizations.

Capacity planners often use the term “peak average” to refer to an average figure derived by concentrating on peak utilizations.

Peak processor times analysis

The analysis of absolute peaks needs to involve two factors: the *number* of peaks and the *duration* of each peak. Using the same data from the previous charts, the analysis of the largest peak is shown in Figure 8-6 on page 204.

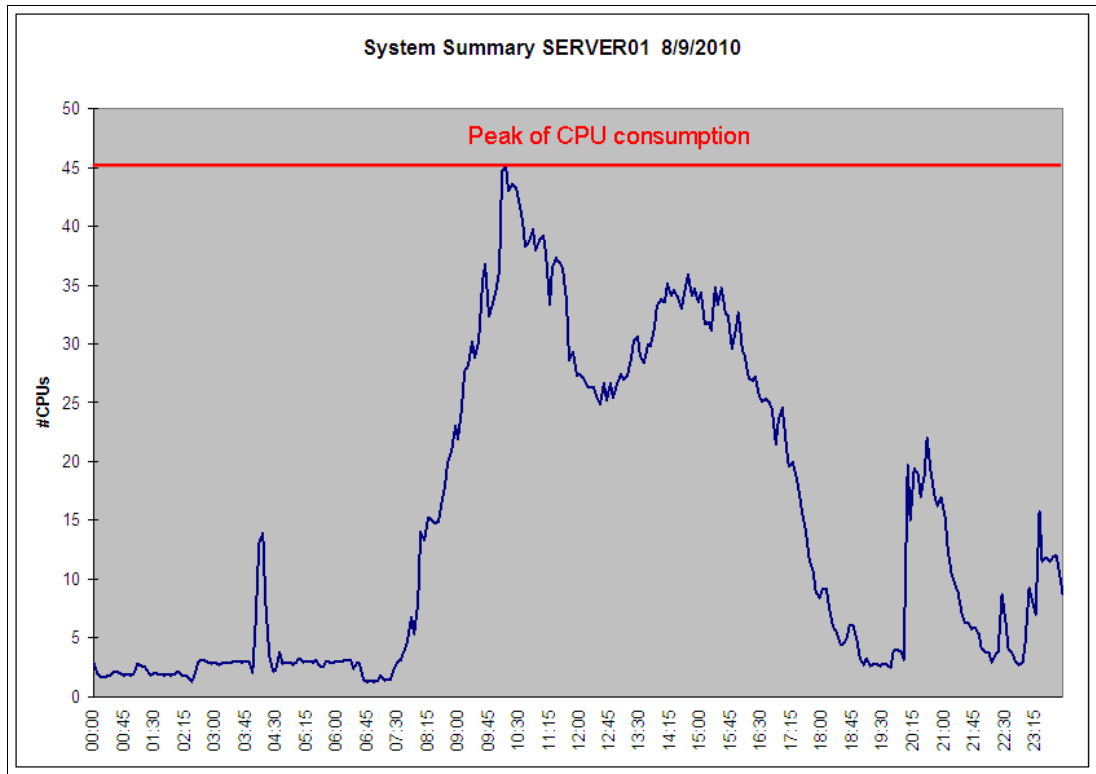


Figure 8-6 Peak of processor consumption

In this example, there is a processor peak around 9:45 am with approximately 15 minutes of duration. 45 processors represent the absolute peak of the interval. Note that in other periods of the day the processor consumption was not even close to this number. Therefore, the server is less busy outside the duration of the absolute peak.

What defines whether you need to use this absolute peak to define the consolidation or migration resources will depend on whether the capability of handling this absolute peak is defined as mission-critical by the client. If the client has a business need to attend the peak as it is defined, then the target system will need to provide resources to accommodate it.

However, in the real world, many times the absolute peak can be disregarded and the planning can be based on a percentage of the absolute peak, or based on another sample altogether. A common criteria is to use the 90th percentile, as discussed earlier in this chapter.

Figure 8-7 shows the stacked processor utilization of three servers. To ensure the proper stacked utilization graph, the samples were gathered at the same date, time, and interval duration. This was done to ensure that the graph represents the peak of the sum for all three servers. In this figure, note that around 10:20 am a peak occurred in all three servers almost simultaneously. By about 11:10 am, all three servers showed a decrease in the consumption of processor.

Using peak average analysis, we can consider the average accumulated processors figure to be around 98, which appears to be a good number that is representative of the exact behavior of three servers when performing together.

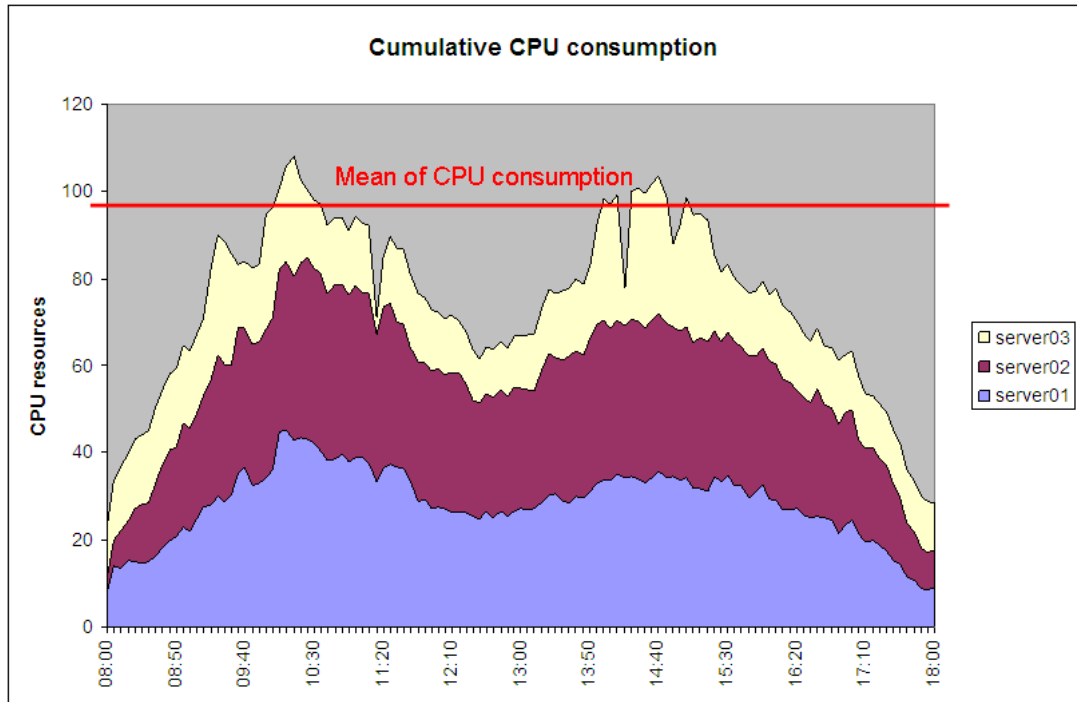


Figure 8-7 Cumulative processor consumption

Overall processor consumption is commonly broken down by most operating systems into at least two component parameters. These are the consumption of resources by the application (USER) and consumption of resources required by the operating system (SYS). Another type of resource consumption parameter is when the operating system is waiting for the completion of an I/O operation (IOWAIT). The (IDLE) parameter shows how much processor resource is still available at any given time.

I/O operations are executed by the operating system and represented as (SYS). Other operations performed by the application are represented as (USER). Figure 8-8 shows an example of daily consumption of resources characterized by the type of consumption.

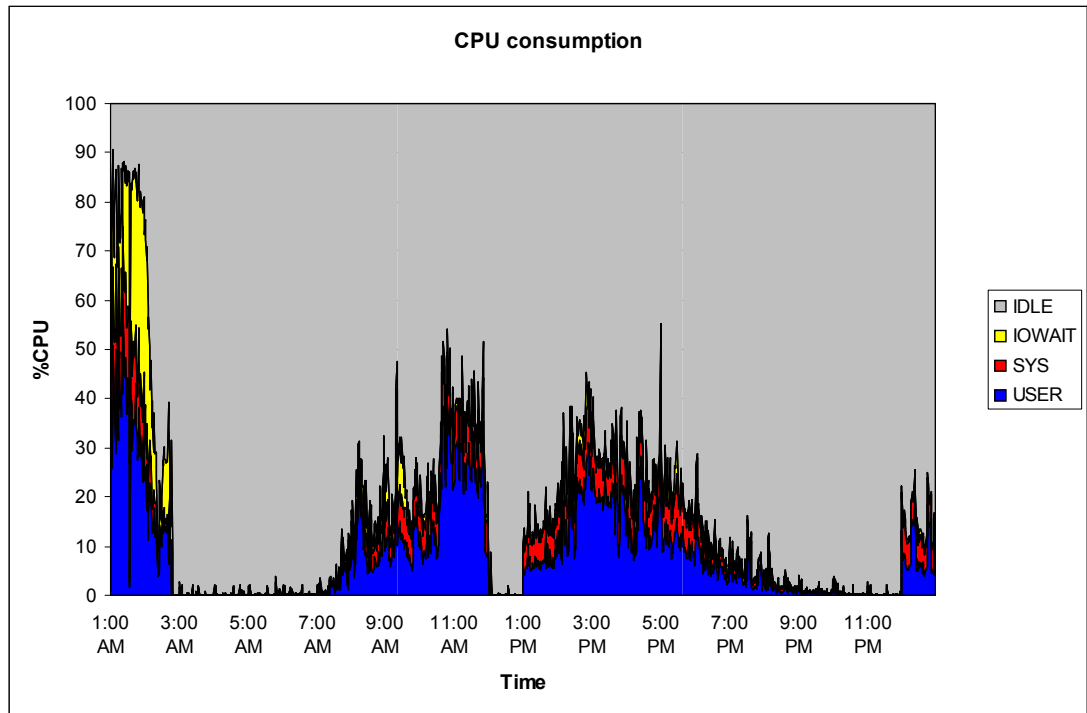


Figure 8-8 Processor consumption by resource type

Resources used by application (USER)

The resources used by the application (USER) will generally not change after consolidation to IBM System z. They represent the actual work being executed in the distributed server.

Resources used by the system (SYS)

Resources used by the system (SYS) are also factored when migrating workloads to System z. The real work is a sum of USER + SYS, and both components need to be sized to run at the target System z. IBM System z has a dedicated I/O subsystem responsible for the execution of all I/O operations. In addition, IBM System z has a special type of processor called the System Assist Processor (SAP) that is responsible for all information exchange between the_CEC⁴ and the I/O subsystem. These facilities and their capabilities are accounted for by selecting the proper workload factor to represent the workload, as previously described.

Resources used by I/O (IOWAIT) and Idle (IDLE)

These resources are not considered when consolidating distributed workloads to System z. IOWAIT is eliminated by the use of the dedicated I/O subsystem that offloads the processors to execute other instructions instead of waiting for the results of the I/O request. IDLE represents white space, therefore not real work, and it should not be considered in the sizing analysis.

Converting distributed server performance to System z MIPS

There is no simple formula that represents the relationship between the performance of RISC and x86-based servers in terms of System z MIPS. IBM uses performance data from a third party company that publishes benchmarks for the RISC and x86 architectures. The relative comparison starts with this data, and then it took years of experience to convert the results to System z MIPS.

⁴ Central Electronics Complex, or, as it is currently referred to with zEnterprise, Central Processor Complex (CPC)

As mentioned earlier, many parameters will influence this conversion, but you need to pay special attention to the workload classification, which will determine the workload factor to be used in the mathematical formula.

Leveraging the expertise of IBM Techline professionals will provide you with the proper sizing and TCO studies.

8.4.4 Memory analysis

When you are sizing memory, be alert for the differences in memory management across architectures and operating systems. Distributed systems running UNIX variants or Linux are normally overconfigured in terms of memory due to low price, lack of proper capacity planning, and the use of extra memory as an operating system's buffers and cache. This practice can translate into inefficient memory usage, because the memory defined might be simply sitting idle, waiting to be used in program execution. The operating system might never need to use it for these purposes, and use it only for buffering and caching data.

In System z architecture, there is a central pool of memory that is shared across partitions. Furthermore, z/VM provides another level of virtualization that is not efficiently used if you define memory buffers and caches waiting for a given program to use them. z/VM is efficient in terms of memory management, and it normally requires only a fraction of the real memory allocated in distributed systems.

If you have access to the actual utilized memory (instead of configured memory), that number is the starting point. Then, you must consider how to virtualize memory, such as which overcommit ratios should be used for a production environment or for a non-production environment and whether you need to be less aggressive in virtualizing memory for applications such as web servers or database servers.

Figure 8-9 shows an example of the efficient use of memory.

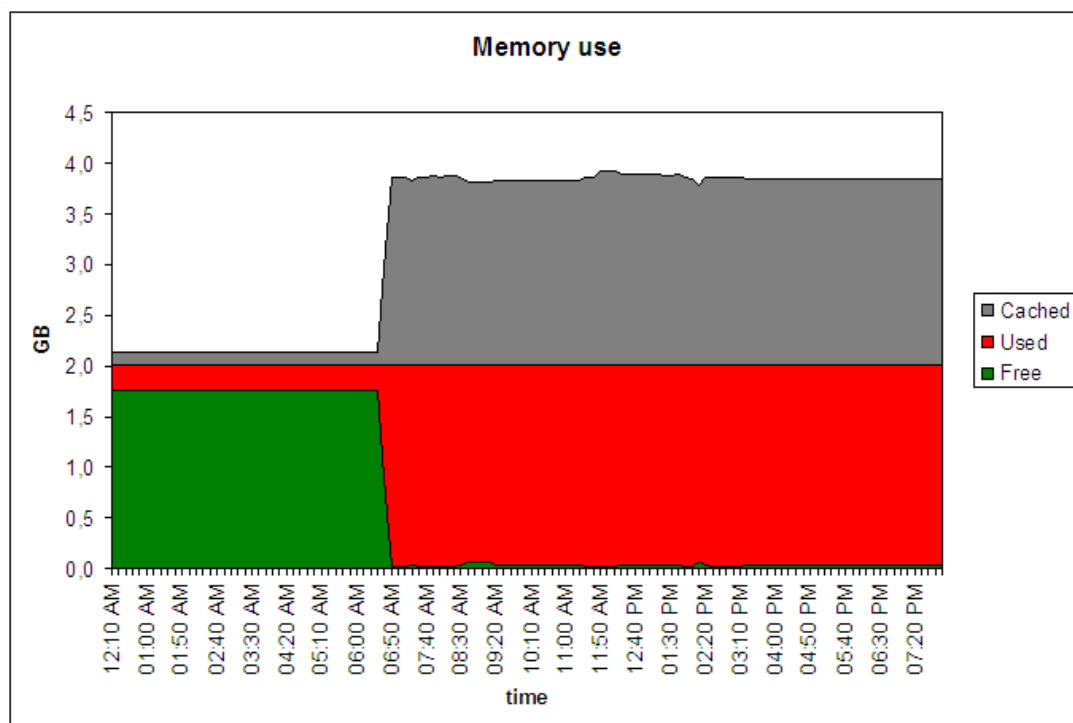


Figure 8-9 Sample of memory usage in a Linux environment

The server in the example is using and reserving 2 GB of virtual memory for program execution. If you add the memory USED to the memory FREE, you will always have the same figure of 2 GB. However, after 6:50 am, there is an abrupt increase of the CACHED virtual memory of up around 1.8 GB, while the server is really using around 168 MB of real memory.

On a z/VM hosted Linux system, you need to keep the Linux guest virtual machine size at a value that does not result in memory being allocated to buffer and cache, so beware of simply adding extra virtual memory without good reason in this environment. In this way you keep all the memory installed on the System z server for program execution purposes in the z/VM Hypervisor pools, and not effectively waste it on Linux guest buffer and cache pools.

8.4.5 Disk analysis (I/O)

The disk analysis should be executed based on I/O activity reports (usage and accesses). For instance, SAR reports can be used to measure the current channels' occupation, the need for adding channels, the use of multipath, and the use of fail-over mechanisms.

A unique differentiator of the IBM System z platform is that it has a dedicated I/O subsystem, which acts as a “broker” in charge of all I/O operations. This feature significantly increases the System z I/O capability because it offloads processor cycles from the main processor to the dedicated I/O subsystem. Another factor influencing System z I/O bandwidth is the number of I/O cards available in the system.

Modern IBM system processors offer FICON channel cards of up to 8 Gbps, where each card has four physical ports with 8 Gbps each.

For the purpose of estimating the I/O bandwidth requirements of the target System z server, you must assess how many I/O cards are required to meet the workload demand. This estimation is a factor of how many servers are being considered in the consolidation study, how heavily these servers are actually using the available bandwidth, and the real speed of the available channels, not just their theoretical capabilities.

Figure 8-10 shows an example analysis created by using the NMON tool.

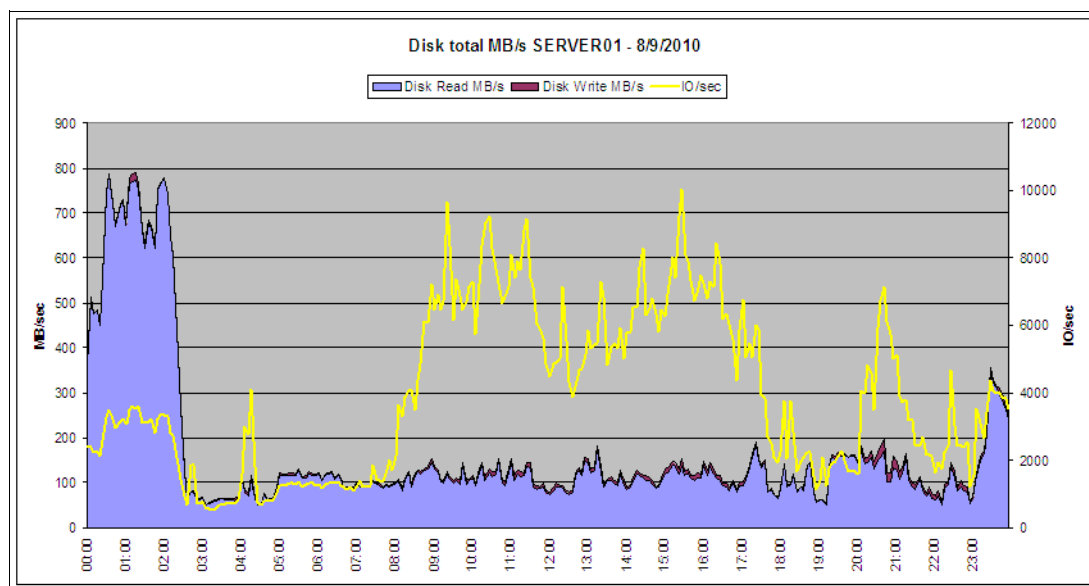


Figure 8-10 NMON disk analysis

This example shows about 100 MB per second (MBps) of bandwidth occupancy during business hours, and 750 MBps of occupation in the backup window. These are expected numbers, because most systems will not need a high level of I/O bandwidth during the day. The numbers of I/Os per second (IOPS) fluctuated around 9000 during the day.

Analyzing Figure 8-10 in isolation, you can say that a single FICON card will suffice and meet the demand because the most recent implementations of High Performance FICON for System z (zHPF⁵) deliver up to 770 MBps of throughput for full-duplex data transfers per channel. In one FICON Express8 card, there are 4 channels. Other enhancements in read and write operations further improve performance.

Note: Keep in mind that these numbers are for a single server. The same task must be performed for all servers being migrated to the IBM System z.

You can use open storage systems with IBM System z as, for example, a Storage Area Network (SAN). Access is through regular FICON channels, but using a different protocol, such as the Fibre Channel Protocol (FCP). After FCP⁶ is defined, you must also analyze the data and estimate the usage for this type of connection. Performance studies suggest high throughput levels when using FCP channels.

Using existing DASD

Existing System z storage (DASDs) can be used, but you might need to allow for the addition of new channels to ensure that new workloads do not interfere with existing workloads. You can further restrict z/VM and prevent its access to z/OS disks, which is generally recommended.

Adding new DASD

Whenever new DASD is added to the existing I/O environment, you must also consider whether there is a corresponding need to add new channels. This analysis involves checking the existing I/O channel utilization levels, defining the new DASD requirements, assessing a possible need to include extra channels to accommodate the new DASD, determining the impact triggered by the addition of new channels, and assessing risk.

To properly assess an existing IBM System z environment, it is best to use the official IBM capacity planning tools and methodologies such as zCP3000, zTPM, and zPCR.

DASD response time

For the majority of disk subsystems (DASD, in the System z case), the ideal response time is below 1ms. The total response time from a disk is represented by the following equation. Expected values are typically less than 1 ms. The response time of the disk is determined by the following sum:

$$DASD_{RT} = DASD_{CT} + DASD_{DT} + DASD_{FP} + CU_{QT}$$

where:

- | | |
|--------------------------|--|
| DASD_{RT} | This refers to DASD response time. The I/O response time is the service time plus any queuing delays. It is calculated from the service time and the average I/O request queue length. |
| DASD_{CT} | This refers to DASD connected time, which is the average time, in milliseconds, during which the device was connected to the channel path while handling an I/O operation (receiving commands or transferring data). This time also includes SEARCH time for DASD. |

⁵ For zHPF enhancements see http://www-03.ibm.com/systems/z/hardware/connectivity/ficon_performance.html

⁶ http://www-03.ibm.com/systems/z/hardware/connectivity/fcp_performance.html

DASD_{DT}	This refers to DASD disconnected time, and is normally equal to 0.0 ms. Abnormally high “disconnected” times can be caused by an overloaded channel path that led to many reconnect misses, or, for DASD, by long SEEKs due to I/O to data at opposite ends of the device.
DASD_{FP}	This refers to DASD function pending. It is the average time, in milliseconds, during which an I/O operation remained pending in the I/O subsystem during the last measuring interval due to path busy conditions.
CU_{QT}	This refers to control unit queue time, which is the time when the device was logically disconnected from the channel subsystem while it was busy with an I/O operation initiated from another system.

If you begin to see response times greater than 10 ms, then you need to understand the issue better. A “health check” in the storage subsystem is required in areas such as cache usage, I/O controllers, transactions competing for the same resources, among others.

8.4.6 Network analysis

Increasingly, the efficiency of the network infrastructure is becoming an essential element to ensure the proper functioning of interconnected equipment. Today almost everything navigates through a network, from voice travelling in cellular networks or voice over IP reaching our computers and dedicated SIP devices, to our televisions, where broadcasts reach our homes through cable TV or the Internet. IBM System z is similar; this modern computer can be interconnected and become part of a client network (intranet), the Internet, and private networks, as well.

Information processed by IBM System z usually arrives through its channels connected to the network or Open Systems Adapter (OSA) cards. Correct OSA sizing is required to ensure good performance under intense data exchange without interruptions or delays.

To be part of this interconnected world, IBM System z requires specialized adapter cards, called “Open Systems Adapter” cards, or OSA for short. A proper determination of the OSA requirements is essential to ensure good performance and minimize interrupt related delays when under stress.

IBM System z OSA channels are rated from 1 Gbps to 10 Gbps. Each channel has a complex and independent processing system that includes the processor, memory, and operating system. However, the intelligence built into OSA controllers does not eliminate the need for you to perform careful network analysis before migrating any workload to IBM System z.

We started with the idea that the migration will consolidate distributed servers into Linux on System z. Next we describe migrating from Linux workloads running on distributed systems to Linux on IBM System z, and explain how to perform the networking analysis.

Network data collection and data analysis

As with other parameters involved in the sizing process, to better analyze the network you need to start with a data collection step. Network data collection can be performed using tools such as SAR or NMON, as previously mentioned. In this section we explain how to use these tools.

Network data collection using the SAR tool

To perform network data collection using the SAR tool in a Linux environment, follow these steps:

1. Install the SYSSTAT package on the host platform.

```
# rpm -ivh sysstat*.rpm
# rpm -qa |grep sysstat*
sysstat-7.0.2-3.el5_5.1
```

2. The sysstat process is called by CRONTAB process, so verify whether you have a file in the /etc/cron.d/ directory.

```
# ls -l /etc/cron.d/
-rw-r--r-- 1 root root 192 Jul  1 13:45 sysstat
```

3. There are two entries in the /etc/cron.d/sysstat file. One entry runs the system activity every 10 minutes, and the other entry generates the daily report.

```
# cat /etc/cron.d/sysstat
# run system activity accounting tool every 10 minutes
*/10 * * * * root /usr/lib64/sa/sa1 1 1
# generate a daily summary of process accounting at 23:53
53 23 * * * root /usr/lib64/sa/sa2 -A
```

4. Define the data collection schedule.

5. Produce the SAR report.

```
# sar -n DEV
```

Example 8-1 shows what a SAR report layout looks like.

Example 8-1 SAR report

```
[root@servername ~]# sar -n DEV
Linux 2.6.18-194.11.1.el5 (servername.ihost.com)      10/21/2010
```

	IFACE	rxpck/s	txpck/s	rxbyt/s	txbyt/s	rxcmp/s	txcmp/s	rxmcs/s
12:00:01 AM	lo	2.65	2.65	198.12	198.12	0.00	0.00	0.00
12:10:01 AM	sit0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12:10:01 AM	eth0	0.15	0.04	21.02	7.40	0.00	0.00	0.11
12:10:01 AM	hsi0	8.42	9.12	4330.29	7340.52	0.00	0.00	0.00
12:20:01 AM	lo	2.55	2.55	137.29	137.29	0.00	0.00	0.00
12:20:01 AM	sit0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12:20:01 AM	eth0	0.17	0.03	20.70	4.14	0.00	0.00	0.15
12:20:01 AM	hsi0	5.60	5.94	1877.44	3525.85	0.00	0.00	0.00
12:30:02 AM	lo	2.63	2.63	142.90	142.90	0.00	0.00	0.00
12:30:02 AM	sit0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12:30:02 AM	eth0	0.16	0.05	21.10	7.53	0.00	0.00	0.11
12:30:02 AM	hsi0	5.86	6.19	2003.73	3635.37	0.00	0.00	0.00
12:40:01 AM	lo	3.29	3.29	453.50	453.50	0.00	0.00	0.00
12:40:01 AM	sit0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12:40:01 AM	eth0	0.14	0.06	18.33	6.37	0.00	0.00	0.12
12:40:01 AM	hsi0	2.86	4.69	919.00	2327.94	0.00	0.00	0.00
12:50:01 AM	lo	4.41	4.41	209.71	209.71	0.00	0.00	0.00
12:50:01 AM	sit0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12:50:01 AM	eth0	0.26	0.06	41.51	11.36	0.00	0.00	0.20
12:50:01 AM	hsi0	30.02	48.18	19104.25	79293.30	0.00	0.00	0.04
01:00:01 AM	lo	2.53	2.53	142.13	142.13	0.00	0.00	0.00
01:00:01 AM	sit0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01:00:01 AM	eth0	0.20	0.10	29.91	16.32	0.00	0.00	0.11
01:00:01 AM	hsi0	6.23	7.48	2297.08	7773.02	0.00	0.00	0.00
01:10:01 AM	lo	2.64	2.64	328.62	328.62	0.00	0.00	0.00
01:10:01 AM	sit0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01:10:01 AM	eth0	0.19	0.08	35.43	20.41	0.00	0.00	0.12
01:10:01 AM	hsi0	93.22	176.68	45820.85	364091.00	0.00	0.00	0.00
01:20:01 AM	lo	2.50	2.50	136.45	136.45	0.00	0.00	0.00

01:20:01 AM	sit0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01:20:01 AM	eth0	0.17	0.03	20.82	4.43	0.00	0.00	0.14
01:20:01 AM	hsi0	5.44	5.73	1599.48	3305.33	0.00	0.00	0.00
01:30:01 AM	lo	2.57	2.57	141.65	141.65	0.00	0.00	0.00
01:30:01 AM	sit0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01:30:01 AM	eth0	0.16	0.05	21.12	7.75	0.00	0.00	0.11
01:30:01 AM	hsi0	5.13	5.34	1948.72	3291.52	0.00	0.00	0.00

This example displays SAR command output. During the 90-minute interval when data was collected, the interface with higher data traffic is hsi0, which presents a full duplex average transfer rate of 62 Kbps. Because the OSA channels offer up to 10 Gbps, a single channel will be able to easily accommodate this requirement.

By using the Link Aggregation Control Protocol (LACP), it is possible to aggregate up to 8 OSA channels to work as though they were one, thus producing a theoretical maximum aggregate throughput of 80 Gbps.

Note: The sample shown here is simply an illustrative example. Keep in mind that each case needs to be analyzed individually.

When planning a migration to System z, consider the use of internal networks (HiperSockets) that reside in System z memory. HiperSockets can greatly reduce the utilization levels of OSA channels. For more information about HiperSockets see *HiperSockets Implementation Guide*, SG24-6816, at:

<http://www.redbooks.ibm.com/abstracts/sg246816.html>

Network data collection using the NMON tool

If you need more detailed network analysis, use the NMON tool. Figure 8-11 illustrates an example of using the network analysis tool NMON.

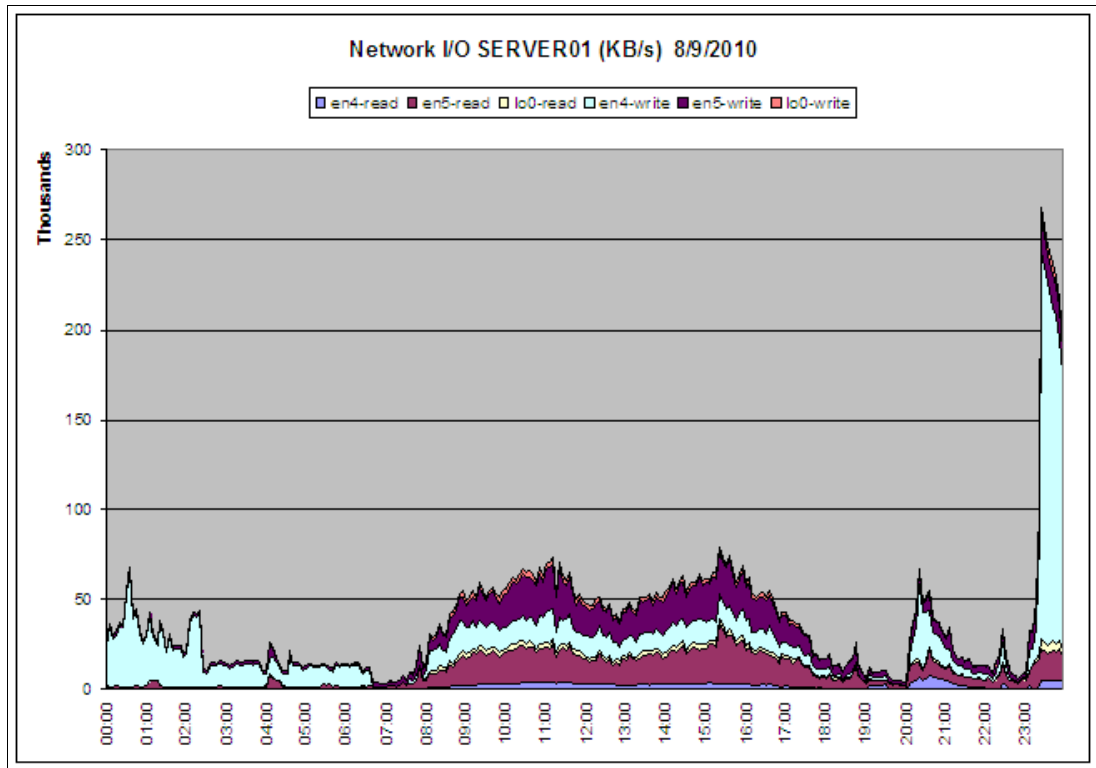


Figure 8-11 Sample of NMON network analyzer

Refer to the appropriate NMON documentation for more detailed information about this tool.

8.5 Sizing methodologies

There are at least two types of sizing you can consider:

- ▶ Sizing for consolidation or migration
- ▶ Sizing from scratch, for IBM middleware or for ISV middleware

8.5.1 Sizing for consolidation or migration

To ensure a reliable assessment of the requirements to enable the consolidation or migration of existing workloads onto IBM System z, you must understand the steps and information required to enable the process. The results of a sizing performed to estimate System z resources to properly run “equivalent workloads” is as good as the input. The better the input, the better the output. Although several of the steps listed here can be considered optional, they are crucial to ensuring the success of any consolidation or migration from workloads running on distributed servers to System z.

Steps for a complete migration project

A successful workload consolidation or migration to System z involves these steps:

- ▶ Completing a server inventory.
- ▶ Defining the scope of migration.
- ▶ Defining the target servers.

- ▶ Configuring the server - performing data collection related to the current environment.
- ▶ Gathering data on current resource consumption (processor, memory, I/O, network).
- ▶ Performing the initial IBM System z sizing.
- ▶ Porting the application development environment to IBM System z (if applicable).
- ▶ Generating test application workloads that are similar or scaled down versions of real production workloads, thereby enabling the assessment, behavioral analysis, and modelling of these workloads as though they were already running in the System z in full production mode. You can use robots or load generators to generate this kind of workload.
- ▶ Performing data collection on IBM System z using CP3KVMXT. TechDocs has specific documentation to guide you through this process. This documentation is available at the following site:

<http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS718>

- ▶ Performing capacity planning and growth analysis on the current System z environment using the zCP3000 tool.
- ▶ Performing pilot testing by bringing up the preproduction environment on the System z and evaluating the initial results.
- ▶ Performing capacity planning and growth analysis on the preproduction System z environment.
- ▶ Performing System z capacity planning report and analysis.
- ▶ Iterating and, if necessary, modifying the initial IBM System z configuration.
- ▶ Performing the production environment definition and build.
- ▶ Becoming ready to migrate the production environment to IBM System z.

In the following sections we explain these steps in more detail.

Completing a server inventory

An inventory of servers is needed to assess which are the most favorable candidates for consolidation on System z. By considering all machines in the landscape, you can verify which servers should be targeted for this consolidation.

Defining the scope of migration

In this case, the term “scope” means establishing the aim of the project, for example making your system highly available, or bringing your environment up to date.

Defining the target servers

When defining the target servers, keep in mind that not everything that runs in a distributed environment is “portable” natively to IBM System z. In this step you define which servers are “friendly” for consolidation.

The zEnterprise solution, consisting of a z196 and BladeCenter® Extension (zBX), can provide a hybrid⁷ environment with applications running on IBM Blades (x86 or POWER7) integrated with the IBM System z196, thus ensuring a high quality of service.

Configuring the server

The data collection process is critical, and is required for each server being consolidated. Table 8-3 lists parameters to collect.

⁷ Statement of Direction - <http://www-03.ibm.com/systems/z/hardware/zenterprise/zbx.html>

Table 8-3 Sample of servers configuration

Vendor	Model/Type	Number of processors (cores)	Clock speed	Memory	L2 cache
IBM	xSeries® 225 (Intel)8649-6BX	1	3.06 GHz	1 GB	512 KB
Sun	Fire F15K	16	900 MHz	24 GB	8 MB
Dell	PowerEdge 430	2	3.20 GHz	1 GB	512 KB

Gathering data on resource consumption

Use the appropriate tools to collect data on resource consumption. Here is a short list of tools available, depending on the operating system you use:

- ▶ Microsoft Windows
 - PERFMON command
 - WINDOWS TASK MANAGER – CPU USAGE HISTORY
- ▶ UNIX/Linux
 - VMSTAT
 - MPSTAT
 - NMOM
 - SAR command
 - TOP command

The time of collection should reflect the typical workload of the environment, around the highest processing period, so collect this data simultaneously on all of the servers involved involved in the migration.

The next step is workload characterization, as discussed in 8.4.2, “Workload characterization” on page 193.

Performing the initial IBM System z sizing

Now you perform the sizing exercise. This step involves estimating the server requirements in terms of processors and memory, thereby ensuring that the allocated resources on the target server are sufficient to sustain preexisting service level agreements.

Table 8-4 on page 215 lists tools that IBM Techline uses and highlights how you can obtain each tool.

Table 8-4 IBM sizing tools

Available planning data	Model with...	Available via...
Processor number	zBOT	IBM employees only
New WebSphere Application Server, WebSphere Portal Server or Apache; rates and sizes estimated	Processor Selection guide (zPSG)	IBM Representative, IBM Business Partner
Current processor LPAR configuration	zPCR	IBM Representative, IBM Business Partner, Client
Server utilization, workload characteristics on UNIX/Intel	RACEv	IBM Rep, IBM BPs through Techline

Available planning data	Model with...	Available via...
Server utilization, workload characteristics on UNIX/Intel	Linux server consolidation (SCON) tool	IBM Techline, IBM RDS
Utilization data for UNIX/Intel	Server utilization reduction facility (SURF)	IBM Techline, IBM RDS
Trans. per second, bytes per second transfer rate, or CCU% utilization of 3745	Communication Controller for Linux (CCL) sizer	IBM Techline
Capacity needed for Linux guests known	z/VM Planner for Linux Guests	IBM Techline
Measurements from z/VM are available (pilot or production)	CP3KVMXT (Data extractor)	IBM Representative, IBM Business Partner
Measurements from z/VM are available (pilot or production)	zCP3000	IBM Techline

Figure 8-12 on page 216 shows the relative positioning of each tool, taking into account graduated factors such as the different methodologies and the “level of accuracy” of the outcome.

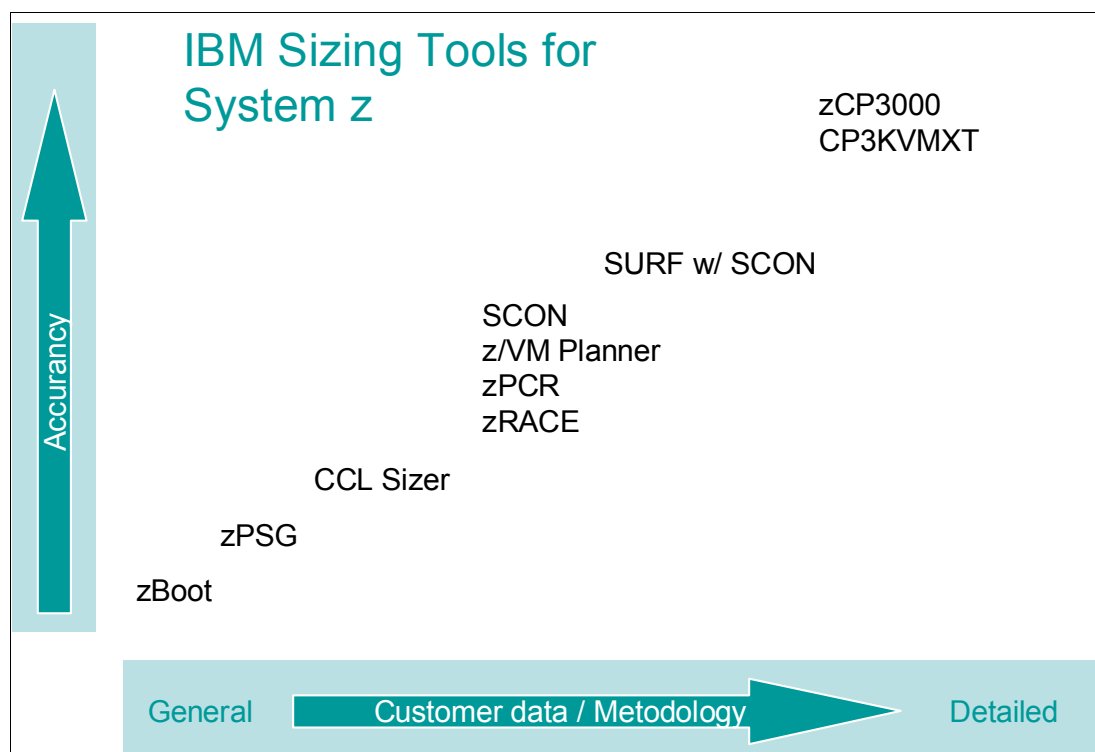


Figure 8-12 IBM internal sizing tools for the IBM System z

Generating test workloads

After you complete the sizing, you can define the initial System z configuration. Using a preproduction test environment can help you to ensure that the initial sizing is realistically close to an optimal final sizing.

Such testing should involve generating workload and simultaneously collecting related performance data on different servers. After the testing is complete and the information has been collected and analyzed, you can verify the “accuracy” of the sizing results.

Be aware, though, that sizing processes cannot guarantee complete accuracy. Although these processes attempt to produce results close to the real needs of an enterprise, real world workloads might not be fully known and performance data might not be available for all servers, and so on. Such factors can increase the number of assumptions required and as a consequence, increase the margin of error.

Using z/VM tools, you can check the actual use of each subsystem being hosted in the system, from the cryptographic coprocessor to the FICON channels.

After the sizing effort is finished, capacity planning for the future should be done, taking into account both organic growth and new projects. Typically, the period covered by capacity planning is three years.

After the sizing and capacity planning studies are complete, you will have the information needed to define the System z configuration designed to meet the demand specified by the current workloads and future growth. The configuration work must be performed by IBM. As previously discussed, it needs to account for the processor needs, memory, and the I/O bandwidth to achieve the business goals of the enterprise.

8.5.2 Modeling and sizing tools

Important: The sizing and capacity planning tools discussed here are tools developed and owned by IBM. The concepts embodied in each tool are explained here, but the sizing study itself must be conducted by IBM.

z Processor Capacity Reference

The z Processor Capacity Reference (zPCR) tool provides capacity planning insight for System z processors running workload under z/OS, z/VM, and Linux when using various LPAR partition configurations. The built-in LPAR analysis capability projects capacity expectations for individual partitions and for the LPAR host as a whole.

Capacity information derived from other sizing tools, such as zPSG and others we describe in this book, can be merged into the model as an additional partition into an existing host and the new, revised capacity requirements taking into account all LPARs can be analyzed. The LPAR Capacity Comparison Report provides the capacity perspective for a planned new partition relative to the capacity being consumed on the current processor to run this workload.

System z Server Consolidation

System z Server Consolidation (SCON, or zSCON) tool provides an estimate of the mainframe capacity to produce an equivalent amount of work currently being executed by distributed systems. It uses what is called the *workload factor* to convert amounts of work from one platform to the other. The workload factor is workload-dependent, therefore it is paramount to properly classify the workload that is under study.

In addition to the make, model, clock speed, and workload peak utilization when running on the existing distributed servers, the zSCON questionnaire asks for additional information about workload characteristics such as the number of concurrent users during peak, whether throughput is gated by processor resource contention or I/O resource contention, and whether the workload has tight or wide address reference patterns.

Another significant factor that the methodology considers is the timing of when the various servers peak. Because System z manages multiple workloads so effectively, server consolidation works well, especially in cases where workloads tend to peak at different times. The System z server consolidation tool also considers whether or not the consolidated environment will run natively or under z/VM.

Communication Controller for Linux

The Communication Controller for Linux (CCL) sizer tool estimates the System z resources required to run the Communication Controller for Linux product with equivalent function as an existing 3745 Communication Controller. Communication Controller for Linux on System z is a software product that emulates 3745 hardware.

The CCL sizer tool is based upon benchmark performance tests with defined workload characteristics. The sizer considers transactional SNA Network Interconnect (SNI) and Boundary Function (BF) workload types. The workload utilization from the source 3745 controllers can be expressed in terms of peak transactions per second, peak bytes transferred per second, or peak percent Central Control Unit (CCU) utilization.

Right-fitting Applications into Consolidated Environments

Right-fitting Applications into Consolidated Environments (RACE) has two “flavors”:

- ▶ RACEv (virtualization), which handles Linux on System z
- ▶ RACEzOS, which handles the z/OS environment.

RACE is a methodology that is implemented through a spreadsheet-based tool, where the sizing and the IT Cost and Value assessments are done together, or any external sizing data can be used to generate only the financial model. Through its TCO analysis (3- or 5-year analysis), RACE helps IBM clients to decide on which platform a given workload can run at lower cost.

Often a TCO study follows a Fit for Purpose (F4P) workshop. In the F4P workshop there is an evaluation of possible alternatives and of how each alternative ranks against a set of functional and non-functional requirements. After the F4P workshop results are known, the client may ask for a TCO (Cost and Value) study to understand the “economics”. When cost is spread across time and it includes operational costs, maintenance costs, and environmental costs, there is a strong chance that the server that “costs less” in terms of acquisition price (or TCA) will not offer the best value (TCO).

System z offers excellent benefits when compared to WebSphere or DB2 on Intel or UNIX platforms. Specialty engines (processors) such as IFLs, zIIPS, and zAAPs are examples of the System z commitment to offer a platform able to compete for new workloads. Note that the use of specialty engines will likely lower software costs. However, the specialty processors can be used only with IBM-specific approved workloads.

RACEv and RACEzOS use a combination of assumptions, estimates, industry standard numbers and actual client data to develop the cost models. But the most important input is the one provided by the user, who can and should override the default costs to tune the model to a specific client’s situation.

Performing data collection, capacity planning, and growth analysis

You can use the following tools to perform data collection, capacity planning, and growth analysis on IBM System z.

- ▶ zCP3000 tool

zCP300 is a tool designed to carry out capacity planning for IBM System z and IBM-compatible S/390® processors running various SCP/workload environments. zCP3000 plans “z-to-z” migrations.

The zCP3000 zVM extract utility (CP3KVMXT) is a zVM-based utility that extracts data from the z/VM monitor and builds the *.edf file that is used and the input into the zCP3000 tool. The *.edf file feeds the zCP3000 with key performance-related information captured by the z/VM monitor.

Note: The z/VM monitor is a no-charge feature that is part of z/VM.

► CP3VMXT tool

The CP3VMXT tool also allows user-defined workload groups, such as LINUXDB or LINUXWEB, to group multiple, related Linux guest machines into one workload. In this case a single growth rate can be applied to the entire group of workloads. The insight gained from the combination of the CP3KVMXT and zCP3000 tools can clearly illustrate whether an environment has adequate resources (IFLs, Memory, or I/Os) to accommodate the various growth scenarios being considered, such as additional workload needs due to the projected growth, the addition of a new workload into the existing environment, or a large number of users.

Working together, zCP3000 and CP3KVMXT enable you to evaluate environments where, for example, there are multiple Linux guests running under z/VM.

In zVM, the term “workloads” is used to refer to groups of zVM user IDs that represent different business processes that can grow at different rates. The zCP3000 can handle this differentiated growth requirement.

8.5.3 Sizing from scratch

In the case of completely new workloads, a sizing from scratch may be needed. Because the applicable workload is not yet running on any platform, there is no current information available regarding its performance, the required capacity, or processing characteristics. New workloads may exploit certain types of middleware, such as WebSphere Application Server or other middleware.

Sizing a Linux on System z or z/OS workload from scratch is not a trivial task, and it requires specialized knowledge. The IBM Techline organization can help you by supplying the skills required to execute this kind of sizing. These specialists work with you to fill in a questionnaire listing the workload’s operational characteristics, and guide you through the entire sizing process. For middleware other than that discussed here, contact IBM directly or an IBM Business Partner for sizing support.

Middleware for Linux on System z

The following section discusses middleware-specific sizing methods for a Linux on System z environment.

Sizing WebSphere

The input parameters required to size WebSphere vary for each individual middleware product that is part of the IBM WebSphere family. The required parameters for each product are listed in the specific questionnaire supplied by the IBM Techline. Among the common required input parameters we highlight the following:

- Number of transactions per second
- Number of users allowed

- ▶ Number of simultaneous connections
- ▶ Whether SSL will be used
- ▶ Use of HTTP server

The appropriate questionnaires are made available by IBM Techline, free of charge. To obtain a copy, contact your IBM Representative or IBM Business Partner and the proper questionnaire will then be sent to you so you can fill it out working with your IBM contact person. IBM Techline will perform the study and discuss the results with you. You can obtain the questionnaire enabling WebSphere Application Server sizing by contacting your IBM Representative.

The questionnaire has two parts. One part asks for minimum data, if no detailed data is available. The second part asks for detailed data. Start with the simpler section first. If required, you will be asked to complete the detailed section.

Sizing DB2

A similar approach applies to DB2 for Linux on System z or DB2 on z/OS. You must obtain specific questionnaires from your IBM Representative or IBM Business Partner. You can obtain the questionnaire enabling DB2 sizing by contacting your IBM Representative.

Sizing Lotus Domino Mail

Lotus® Domino® Mail server is an excellent candidate to be run on System z under Linux. IBM itself is a reference in this area. Under the project name “IBM Project Big Green” IBM is in the process of moving 3900 of its own UNIX/Linux mail servers running Lotus Domino to about 40 physical servers running Linux on System z. For more details, see:

<http://www.ibm.com/systems/greendc/resources/info/green20/index.html>

Again, a client fills out the questionnaire with the support of an IBM Representative. The questionnaire is then sent to Techline, which will engage an IT Specialist with the client. You can obtain the questionnaire enabling LOTUS DOMINO MAIL sizing by contacting your IBM Representative.

Sizing Oracle database server

To properly size Oracle database server workloads you must provide the following information:

- ▶ Targeted database server version
- ▶ Number of simultaneous connections
- ▶ Database size in GB or TB
- ▶ Whether or not Oracle RAC is part of the solution
- ▶ Output of the Oracle system command **show parameters**

After the information is collected it must be sent to IBM Techline through your IBM Representative or IBM Business Partner.

Sizing WebSphere MQ

To size WebSphere MQ, you fill out a form with simple information regarding the number of messages processed per second. You can obtain this form from your IBM Representative or IBM Business Partner. After completion, it will be forwarded to the IBM Techline team.

You can obtain the questionnaire enabling WebSphere MQ Series on System z sizing by contacting your IBM Representative.

Middleware for z/OS

Although consolidation is a Linux on System z strength, keep in mind that z/OS also can be the target of workload consolidation from distributed systems to System z. This is especially the case for workloads with extreme requirements for high availability. There are many reasons to choose z/OS as the host environment, including the capability of interconnecting individual System z servers in Parallel Sysplex, solid security, and so on.

Sizing WebSphere Application Server for z/OS

Sizing WebSphere Application Server for z/OS also requires a questionnaire to be filled out and submitted to IBM Techline. The questionnaire is the same one as used for WebSphere Application Server sizing for the Linux on System z environment. However, when filling out the questionnaire targeting z/OS as the host operating system, keep in mind z/OS environment specifics.

You can obtain the questionnaire enabling WebSphere Application Server sizing by contacting your IBM Representative.

Sizing DB2 for z/OS

Sizing DB2 for z/OS requires a questionnaire where information such as number of transactions per second and others related data is gathered.

You can obtain the questionnaire enabling DB2 for z/OS sizing by contacting your IBM Representatives.

Sizing Parallel Sysplex

Sizing a Parallel Sysplex is not a trivial task, and it cannot be accomplished by filling out questionnaires. To size a Parallel Sysplex environment, contact and engage the IBM Techline.

Sizing SAP solution for IBM System z

IBM is able to run a complete SAP infrastructure entirely on System z. To size an SAP Solution for IBM System z, contact your IBM Representative or IBM Business Partner.

You can obtain the questionnaire enabling SAP Solution for System z sizing by contacting your IBM Representatives.

8.5.4 Sizing High Availability and Disaster Recovery

High Availability (HA) and Disaster Recovery (DR) are important areas to pay attention to when producing valid sizing, capacity planning, or TCO reports. We elaborate on these aspects in other chapters in this book.

Philosophical differences exist regarding how to architect these solutions. In the distributed world, HA calls for two servers for each function, whereby one server assumes the active role and the other server assumes the hot-standby role, becoming active in case of a failure on the main server. However, in the mainframe world, given the highest system (hardware and software) availability, HA and DR are implemented through dormant processors that can be activated when the need arises.

As an example, consider a scenario where two IBM System z servers are running in active/active mode and you need to plan the capacity of each server in a DR event. In this scenario, one server should be capable of assuming of the other server's workloads, while also running its own workloads.

At this point you must measure the amount of memory, the number of channels, and how many processors are needed to absorb all workloads running in two System z servers in a single System z server.

Figure 8-13 illustrates this scenario. It shows two sites working together using Parallel Sysplex and DB2 Data Sharing. The disks are shared between the two sites using Peer to Peer Remote Copy (PPRC) technology. If Site A is lost due to an external event, then LPAR 1 must come up in Site B. Therefore, in Site B you will have two LPARs in total.

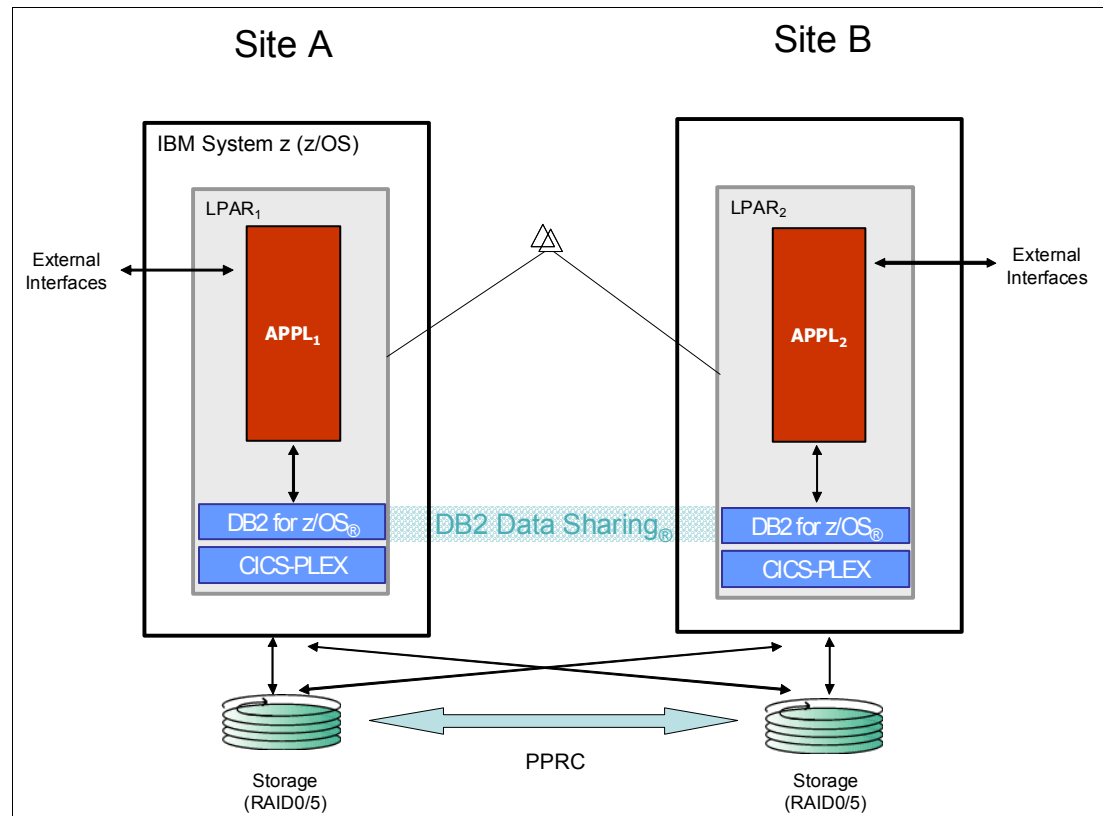


Figure 8-13 Sample of HA scenario with IBM System z

Bringing up this second LPAR in Site B will not be a problem if there is sufficient capacity (memory, processor, and channels) to support two production LPARs on a single site. You need to consider the following aspects when planning:

- ▶ Planning capacity on demand in both sites
- ▶ Planning for memory needed for both sites
- ▶ Planning for channels needed for both sites

Planning capacity on demand

IBM System z offers various alternatives in terms of capacity on demand, such as On/Off Capacity on Demand, Capacity for Planned Event, and Capacity Backup. This section briefly examines each of these alternatives.

On/Off Capacity on Demand

On/Off Capacity on Demand (On/Off CoD) allows you to temporarily enable unassigned CPs, IFLs or zAAP, zIIP or ICF specialty engines that are physically available in the current

machine or model, or to change capacity settings of the existing active CPs to increase their capacity to meet peak workload demands.

Capacity for Planned Event

Capacity for Planned Event (CPE) is offered with IBM System z to provide replacement backup capacity for planned downtime events. For example, if a server room requires repair work, replacement capacity can be temporarily activated on another z196 in the client's environment. That will ensure production is not disrupted due to the server room repair work.

CPE usage: CPE is designed for planned replacement capacity only. It cannot be used for peak workload management.

Capacity Backup

Capacity Backup (CBU) provides reserved emergency backup capacity to be used during unplanned situations in which capacity is lost in another part of the enterprise and there is a need to recover the initial capacity by adding the backup capacity available on designated IBM System z servers.

CBU is the temporary activation of PUs and it is offered as follows:

- ▶ For up to 90 contiguous days, in case of a loss of processing capacity as a result of an emergency or disaster recovery situation
- ▶ For up to 10 days during testing disaster recovery procedures test (additional tests can be purchased from IBM)

CBU usage: CBU is designed for disaster recovery (DR) purposes only. It cannot be used to accommodate peak workload management or planned events. CBUs must be configured to support the DR plan in place at the client's enterprise.

Memory planning

In a disaster recovery event, you need to dimension the System z memory with enough capacity to support all of a client's production LPARs.

Channel planning

In a disaster recovery event, make sure that your channel planning accounts for sufficient resources to avoid I/O contention when all production LPARs are running at a single site. Because System z is generally configured with enough I/O bandwidth, channel planning is not likely to become a problem. However, pay attention to DR channel management enablement where I/O devices need to be accessible across sites.

Note: Leveraging the extensive experience and capabilities of IBM Techline can help you achieve a reliable DR and HA design. If you do not have direct access to IBM Techline, contact your IBM Representative or IBM Business Partner.

8.6 Capacity planning methodologies

Depending on the available time and skills, acceptable cost, and level of detail and precision required, you can choose among several different methods when executing a capacity planning study. This section outlines and assesses those methods to help you choose the one that is right for you.

8.6.1 Guidelines

A *guideline* can be defined as the merger of an educated guess and a rule of thumb. It is simple, so it enables informal projections with no documentation, simply based on previous experience and knowledge of the system. An example of a guideline is when historical workload utilization is plotted in a graph against time, and a straight line following the tendency is used to predict future utilization.

Using this approach, additional CP resources are added to the system when necessary, or workloads are moved to adjust the environment to the needs. Pen, paper, and calculator are often the tools used to create guidelines for capacity planning methods. Accuracy cannot usually be reliably predicted with such methods.

8.6.2 Linear projections

Linearization is normally not a useful practice in capacity planning, because it does not account for the Symmetric Multiprocessing (SMP) architecture of modern computers. A linear projection can be done using almost any tool, from simple diagrams based on spreadsheets to more sophisticated methods. Tools such as zCP3000 and the System z Processor Capacity Reference (zPCR) are based on LSPR data. However, LSPR data is not linear because it curves down when the number of processors increases. This is just physics, and the LSPR is true to the SMP architecture. These tools are examples of those that can help you estimate various configurations and workloads using a “spreadsheet-like” approach to plot results. However, linearizing projections are not particularly helpful in capacity planning, especially if the target environment has a large number of processors.

8.6.3 Analytic methods

An *analytic method* is fundamentally based on mathematical calculations derived from queuing theory. Projections based on analytic models can provide an accurate insight into forecasting values such as processor utilization, response time evaluation, capture ratios, effect of buffering, and the interdependent effects of queuing within the system. Analytic methods take an input from data gathered from existing environments and then use mathematical formulas to come up with projected resources required to satisfy the target scenario, normally after the effects of additional workloads and the growth forecasted for the existing workloads are accounted for. The models can be useful for projecting real world situations because they allow the study of “what if” scenarios. zCP3000, zMCAT, and zVM Planner are examples of many IBM tools that fall into the analytic methods category.

8.6.4 Discrete methods

A *discrete method* is an application of discrete simulation. Unlike analytic simulation, discrete simulation is not based on mathematical formulas. Using discrete capacity planning methods, dissimilar workloads and their effect on each other are modelled. An example is the effect of combining batch and interactive workloads on the same z/OS system.

A discrete simulator does what its name suggests: it simulates events that take place in a system, such as the input of a message to the system, the message arrival at the processor, the instructions required to process it, polling the terminal, and sending the response back. All these events are simulated, tabulated, and reported by the discrete simulation model. Sometimes a discrete simulation requires more detailed input data than an analytic simulation.

8.6.5 Real world benchmarks versus standard benchmarks

Performance on System z is normally expressed in MIPS, or millions of instructions per second. However, the best way to express mainframe performance is by using the Large Systems Performance Reference (LSPR), where new generations are compared to older generations and expressed in terms of the percentage of improvement against a base model. More about LSPR can be found at the following site:

<https://www.ibm.com/servers/resourcelink/lib03060.nsf/pages/lspindex?OpenDocument>

For various reasons, standard benchmarks used to measure distributed systems are not appropriate for the mainframe. Such benchmarks do not exercise the entire mainframe architecture, because they were designed to run in a different architecture.

System z processors reside within the Central Electronics Complex (CEC)⁸. Real world benchmarking will not run standard benchmark workloads, but real client workloads. Client real workloads are run in suitable System z hardware configurations and the actual processing power, leveraging not just the processor but the entire mainframe architecture, is evaluated. Real world benchmarking is more effective than the previous methods, but also more expensive and time consuming.

There are two types of benchmarks:

- ▶ The client's own benchmarks

The client runs its own workload on a specific hardware configuration, which resembles the anticipated configuration for the target system.

- ▶ Third party benchmarks

You can use evaluations from benchmark studies conducted by consulting companies, and so on.

Unlike the analytic and discrete methods, the benchmark approach requires executable code and installed hardware to predict how a certain combination of resources will perform in a specific situation. This allows analysis of many different configurations and workload possibilities, often with simply a change of a single parameter.

Benchmarks allow measurements of areas such as locking, internal serialization, I/O (DASD, control units, strings), storage, and release-to-release software migrations.

Figure 8-14 illustrates the positioning of these capacity planning methods. It illustrates that there is a relationship between “cost” and the level of detail or accuracy. All methods are shown rising from the x-axis. Methods can, in reality, have an accuracy of as low as 0%. This is the case, for example, if you run a detailed benchmark while neglecting possible bottlenecks in the system.

⁸ In various IBM literature, a CEC is also referred to as Central Processor Complex (CPC).

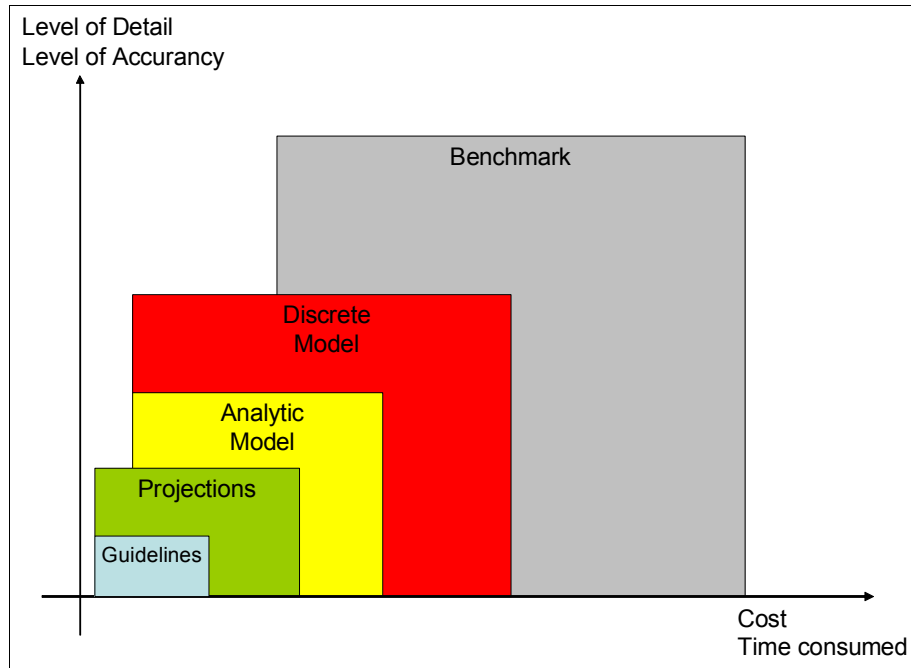


Figure 8-14 Capacity planning methods positioning

8.6.6 Processor capacity planning

Processor capacity planning is carried out by evaluating and analyzing the current used capacity against the projected capacity when, for example, new workloads or annual growth are factored in. The current utilized processor capacity will initially be provided by the sizing model, or actual measurements into existing System z environments. The estimated capacity is defined by the sum of the organic growth plus the growth anticipated by known factors such as increased usage due to new workloads. The estimated capacity is normally expressed as an added percentage (%) to the current utilized capacity.

The methodology allows you to apply differentiated growth rates against each individual workload. For instance, you can define 10% year-to-year growth to be applied against Application A and 50% against Application B. The existing capacity planning tools enable the distinction of each workload, individually.

One way to produce reliable input for a processor capacity planning study is to create a table listing each individual workload being consolidated and indicating the expected associated growth for each workload. Note that in a Linux consolidation project, each workload becomes a separate virtual machine, because it is normal practice with distributed systems. After the table is completed, you can send the information to IBM Techline in conjunction with the data collected for the sizing or capacity planning.

Carry out a review of the current and ongoing processor capacity requirements at least every three months to eliminate or minimize unplanned outages.

8.6.7 Memory capacity planning

Memory capacity planning tends to be simpler than the processor capacity planning. It is less common for existing applications to require significant amounts of extra memory over time, unless there has been a major upgrade such as the introduction of a new function.

However, situations in which completely new applications are introduced, and which necessitate an increase in the system memory size, are commonly encountered. For this reason, all projects that account for the addition of new workloads should be subject to additional sizing and capacity planning.

If there is not enough memory, you can use paging areas or expanded memory.

8.6.8 I/O capacity planning

Perform I/O capacity planning whenever any of the following events occur:

- ▶ New applications are being introduced to the environment
- ▶ New or additional I/O devices are to be introduced
- ▶ There is a verified high DASD I/O response time
- ▶ Changes to the system environment or configuration are planned

If any of these factors arise you need to review the system infrastructure review and to replan the capability of the I/O system. These tasks must be conducted with the help of IBM Techline.

8.6.9 Tools of capacity planning

There are a number of tools that can be used for capacity planning. Many of these are designed for IBM internal use only, but an exception is zPCR, which is available to clients and Business Partners at:

<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS1381>

There is an online education module that is required to enable the installation.

Engaging IBM in your capacity planning allows you to use all the IBM capacity planning capabilities. The IBM Global Techline can help you to perform such studies.

The most complete System z capacity planning tool is zCP3000. As previously noted this tool allows you to use real data and understand the current environment, to properly account for projected growth, and to forecast the requirements to fulfil the needs after a given amount of time.

Figure 8-15 on page 228 shows 30% growth forecast generated by zCP3000.

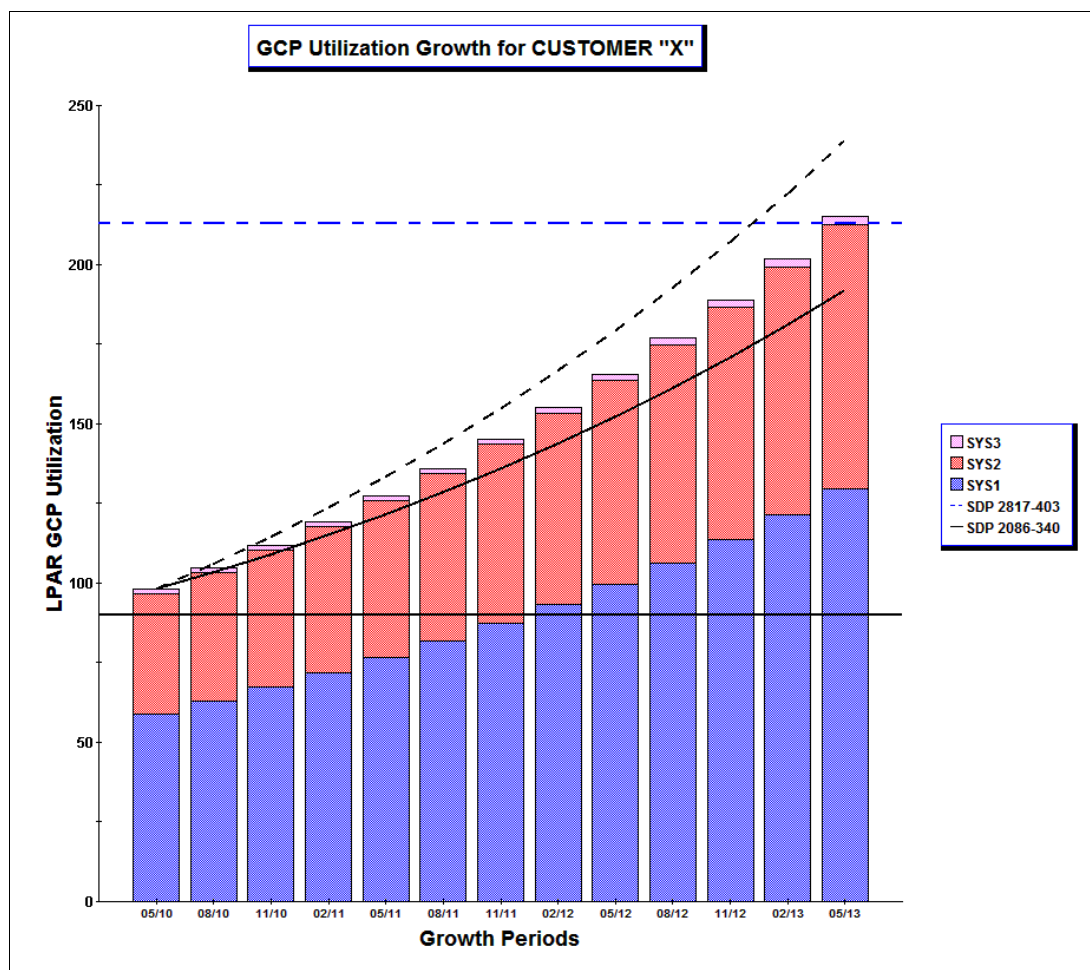


Figure 8-15 Sample of processor growth using zCP3000

You can also undertake a growth forecast yourself by using linear projections or Excel spreadsheet graphics. However, a study performed by the experts at IBM Techline using the zCP3000 tool can produce more accurate results.

8.7 TCO analysis

Total Cost Of Ownership (TCO), also known as “Cost and Value”, is closely related to sizing and capacity planning. After the required configuration that meets the business objectives is estimated, it is often followed by economic considerations with the goal of enabling a better comparison among the alternatives being considered. The Cost and Value discussion also elevates the decision-making process to the client’s CXX (CIO, CFO, CEO) sphere, and can move the process away from technical biases to a more strategic standpoint.

A TCO study produces a cost analysis within a given period of time, normally 3 or 5 years. Because the acquisition cost, known as the Total Cost of Acquisition (TCA), is charged against the first year, normally the initial costs are higher than the cost in subsequent years. This discrepancy in cost distribution is the main reason to justify a TCO study because it illustrates that solutions producing apparent lower costs based only on acquisition may in fact be more expensive when considering the running costs along its useful life including maintenance, energy, space, and other factors.

Studies based on IBM server consolidation engagements demonstrate that the costs of the various elements in an IT infrastructure have changed radically in the past 15 years. In 1995, the following scenario prevailed:

- ▶ 65% - Hardware costs
- ▶ 14% - Software costs
- ▶ 14% - People costs
- ▶ 07% - Other costs

In today's scenario, the changes have shifted considerably:

- ▶ 18% - Cost of hardware
- ▶ 27% - Software costs
- ▶ 45% - People costs
- ▶ 10% - Other costs

This shift in cost elements is illustrated in Figure 8-16.

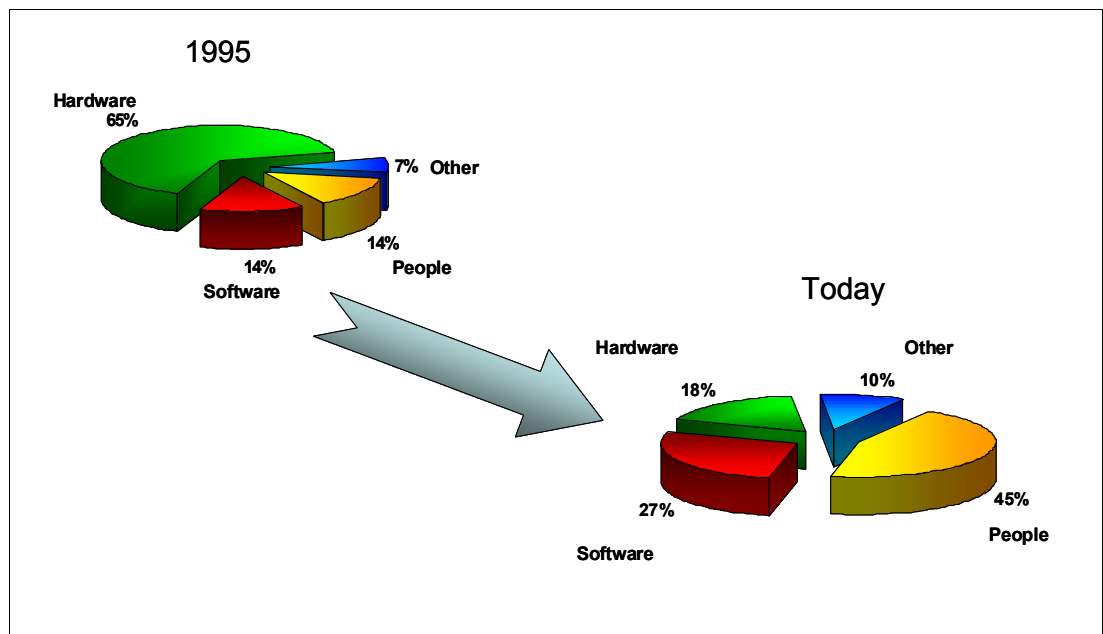


Figure 8-16 Study based on IBM server consolidation engagements

Examining the marketplace today, it is clear that the cost of acquiring new hardware is about constant but the cost related to power and cooling, and mostly the cost related to server management and administration, are all increasing at a large pace. See Figure 8-17 on page 230.

New Economic Model for the Datacenter

Shifts to Automation Tools are a Requirement

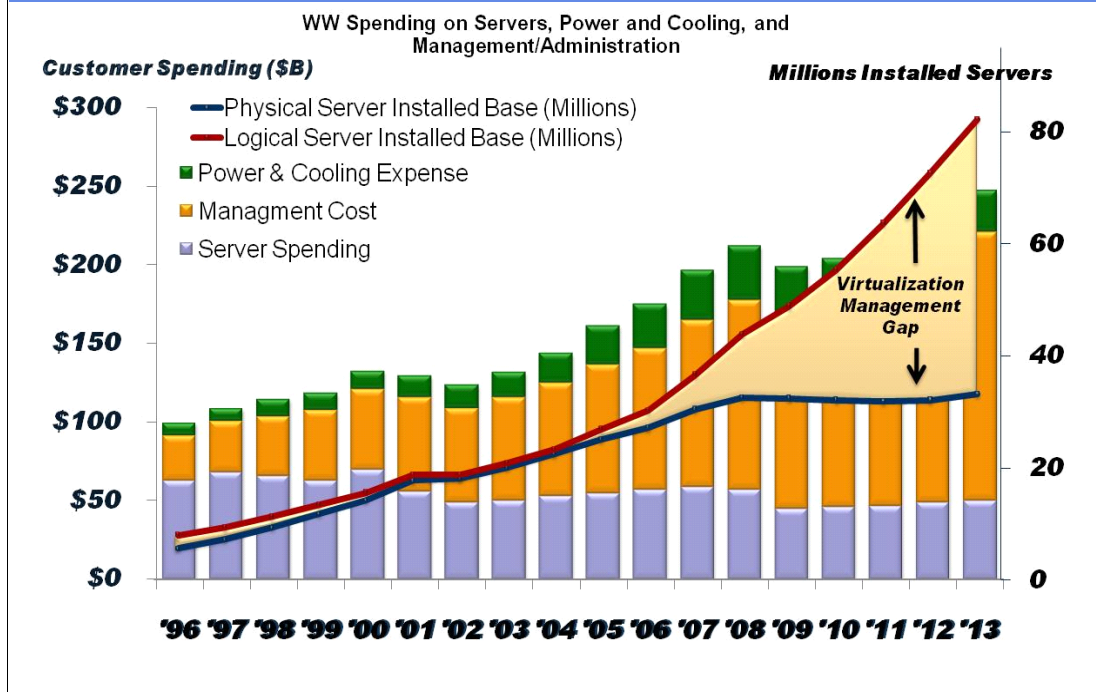


Figure 8-17 IDC study regarding data center costs⁹

A serious TCO study includes, at minimum, the following items:

- ▶ Hardware acquisition costs
- ▶ Hardware maintenance costs
- ▶ Administration costs
- ▶ Software acquisition costs
- ▶ Software maintenance costs
- ▶ Migration costs
- ▶ Space costs
- ▶ Power and cooling costs
- ▶ Disaster recovery costs
- ▶ High availability costs
- ▶ Downtime costs

IBM Techline has tools to perform TCO studies, based on client requirements and the suggested settings for sizing and capacity planning. A TCO study can help you highlight the value that the System z brings to the table. To perform a TCO study, contact your IBM Representative or IBM Business Partner.

8.8 Post-sizing review

After the sizing process is concluded, a report is generated. This is normally done by the person who did the sizing. It is good practice to examine the results and understand the

⁹ Source: IDC, Market Analysis Perspective: Worldwide Datacenter Trends and Strategies 2010, Doc # 227673, Mar 2011

outcome before going to your client. Contact the subject matter expert who performed the sizing to clarify questions you might have, and to explain how the results were produced, which data was considered, what assumptions were made, and so on. A clear link between the current system and future systems needs to be understood. This post-sizing review is an important part of the process because it allows you to grasp the scale of the new server environment so you can then begin to properly plan for the migration.

Use the available IBM technical support resources to help you fully understand the results. If the providing organization was Techline, you can obtain the sizing report, the TCO study, and the final configuration to be included in the proposal. The only other fact to be considered is that any sizing is a guideline. Sizing results may need to be overridden to match the local business needs, as long as the risks are mitigated.

Related publications

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

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 234. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *A Guide to the ACI Worldwide BASE24-eps on z/OS*, SG24-7684
- ▶ *IBM Communication Controller for Linux on System z V1.2.1 Implementation Guide*, SG24-7223
- ▶ *Enterprise Extender Implementation Guide*, SG24-7359
- ▶ *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 3*, SG24-7534
- ▶ *Experiences with Oracle Solutions on Linux for IBM System z*, SG24-7634
- ▶ *Security on the IBM Mainframe*, SG24-7803
- ▶ *OS/390 MVS Parallel Sysplex Capacity Planning*, SG24-4680
- ▶ *IBM zEnterprise System Technical Guide*, SG24-7833
- ▶ *HiperSockets Implementation Guide*, SG24-6816

Other publications

These publications are also relevant as further information sources:

- ▶ *z/VM: Running Guest Operating Systems*, SC24-6115-03

Online resources

These websites are also relevant as further information sources:

- ▶ Benefits z/VM offers a Linux guest environment:
<http://www.vm.ibm.com/linux/benefits.html>
- ▶ The Parallel Sysplex website:
<http://www-03.ibm.com/systems/z/advantages/pso/index.html>
- ▶ For Oracle ASM, visit:
<http://www.oracle.com/technology/products/manageability/database/pdf/asmov.pdf>
- ▶ For details about certification, visit:
<http://www.vm.ibm.com/security/>
- ▶ For specific Linux distributions and kernel levels, go to:
<http://www.vm.ibm.com/related/tcpip/>

- ▶ For a link to SSL Server configuration, visit:
<http://www.vm.ibm.com/related/tcpip/vmsslinf.html>
- ▶ For information about performing data collection on IBM System z using CP3KVMXT, see:
<http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS718>
- ▶ IBM Project Big Green is at:
<http://www.ibm.com/systems/greendc/resources/info/green20/index.html>
- ▶ Tools for capacity planning can be found at:
<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS1381>

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

A

- Access control 163
- Access Control List (ACL) 163
- ACID 154
- Activity levels 24
 - Active / active 25
 - Active / cold 25
 - Active / ready 25
 - Active / trickle 25
 - Active / warm 25
 - Dual active 24
 - Dual active / ready 25
 - Dual active / trickle 24
 - Dual active / warm 25
 - Independently active 25
- address space 59
- address spaces 50
- Advanced Program to Program Communication (APPC) 155
- Atomicity 154
- Authentication 162
- Authorization 162
- authorization layer 84
- Availability 4, 11

C

- capacity
 - dormant 84
- Capacity Backup (CBU) 83, 130
- Capacity for Planned Events (CPE) 83, 130
- Capacity on Demand (COD) 83
- Capacity on Demand (CoD) 130
- CBU (Capacity Backup) 83
- Central Processor Complex (CPC) 50, 124
- Channel Subsystem Priority Queuing 62
- class 63
- Clustering 18
- Clustering, virtualized 19
- common area 86
- Common Criteria 88
- Confidentiality 161
- Configuration Items (CI) 121
- Continuous availability 75
- Continuous Availability (CA) 4
- Continuous operation 5, 75
- Continuous Operations (CO) 4
- control block 60
- control processors (CPs) 53
- Controlled Access Protection Profile (CAPP) 88
- Controlled Access Protection Profile (CAPP), in z/VM 88
- Coupling Facility (CF) 79
- CPE (Capacity for Planned Events) 83
- Cross-System Coupling Facility (XCF) 70

D

- Data Facility Hierarchical Storage Manager (DFSMSHsm) 123
- Data replication 78
- Data set
 - partitioned 106
 - sequential 105
 - Virtual Storage Access Method (VSAM) 106
- Data sharing 78
- Data sharing group 79
- DB2
 - compression 99
 - parallelism 101
 - query CP 101
 - query I/O 101
 - sysplex query 101
 - utilities 101
 - partitioning 100
 - by growth table space 100
 - range partitioning table space 100
 - table partitioned 100
- DB2 data sharing 79, 98–99
- DB2 governor 65
- DCM (Dynamic channel path management) 62
- depth scalability 82
- Discretionary Access Control (DAC) 88
- Distributed Computing Environment (DCE) 155
- Distributed transaction 159
- Domains of Transformation 148
- dormant capacity 84
- Duplexed wave division multiplexer 32
- DVIPAs (Dynamic VIPAs) 70
- Dynamic channel path management (DCM) 62
- Dynamic Cross-System Coupling Facility (dynamic XCF) 70
- Dynamic routing 70
- Dynamic VIPAs (DVIPAs) 70
- dynamic virtual IP address (DVIPA) 69

E

- Enclave 61
- engines 53
- Enterprise Application Integration (EAI) 157
- Evaluation Assurance Level (EAL), in z/VM V5.1 89
- Evaluation Assurance Level (EAL), in z/VM V5.3 89
- Evaluation Assurance Level (EAL), in z/VM V6.1 89

F

- Failures in Time (FIT) 11
- Fault tolerance 5
- fetch bit 86
- File systems
 - Distributed File System 160

- Entry-sequenced 160
- Flat file 160
- Journal File System 160
- Key-sequenced 160
- Flashcopy 80

G

- General Parallel File System (GPFS™) 160
- Geographically Dispersed for Open Clusters (GDOC) 82
- Geographically Dispersed Parallel Sysplex (GDPS) 50, 77, 81
- Global Copy 80
- Global Mirror (PPRC) 80
- Global resource serialization (GRS) 65
- Group Buffer Pools 79
- Guest
 - second level, under z/VM 88

H

- Hardware Management Console (HMC) 53
- Heartbeat 143
- High availability 75
- High Availability (HA) 4
- HiperSockets 69
- HMC 84
- horizontal scalability 82

I

- IBM License Compliance Manager 124
- IBM Tivoli Decision Support (TDS) 117
- IBM Tivoli System Automation for z/OS (SA z/OS) 116
- IBM Tivoli Workload Scheduler (TWS) 117
- IFL (Integrated Facility for Linux) 125
- Infiniband 18
- Information Management System (IMS) 160
- initiators 63
- Integrated Coupling Facility (ICF) 124
- Integrated Facility for Linux (IFL) 124–125
- Intelligent Resource Director (IRD) 61

J

- Java Secure Socket Extension (JSSE) 161
- Job Entry Subsystem (JES) 63

K

- Key Performance Indicators (KPIs) 117

L

- Lightweight Directory Access Protocol (LDAP) 95
- Lightweight Directory Access Protocol (LDAP), in z/VM V5.3 95
- Linux on System z 51
- Linux-HA 143
- Log shipping 105
- logical partitions (LPARs) 53
- loop 65

M

- management application 84
- Mandatory Access Control (MAC) 88
- Mean Time to Failure (MTTF) 11
- Mean Time to Repair (MTTR) 11
- Metro Mirror (PPRC) 80
- Metropolitan cluster 31
- Middleware 157
- Minidisk 88
- Multifiber cable 32

N

- Non-repudiation 163

O

- On/Off Capacity on Demand (On/Off CoD) 83, 130
- Open storage 81
- Open Systems Adapter Express2 (OSA-Express2) 56
- Open Systems Adapter-Express (OSA-Express) 56
 - OSA-Express (Open Systems Adapter-Express) 56
- operating system 59
- Operational models
 - Single Site Parallel Sysplex 131, 133
- Oracle
 - Automatic Storage Management (ASM) 79, 103
 - Cache Fusion 103
 - Cluster Ready Services (CRS) 103
 - DataGuard 104
 - Grid Control 104
 - Maximum Availability Architecture (MAA) 103
 - Real Application Cluster (RAC) 79, 103
 - System Global Area (SGA) 103
- OSA-Express 69

P

- Parallel Sysplex 50, 71, 77
 - components 77
- Peer to Peer Remote Copy (PPRC) 80
- Physical cluster 22
- Planned outage 130
- portsharing 71
- PostgreSQL (PGSQL) 104
- primary router (PRIRouter) 69
- private area 86
- Processin Unit (PU) 124
- Processor Resource/System Manager (PR/SM) 53

Q

- QDIO mode 74

R

- RACF (Resource Access Control Facility) 57
- Recovery Point Objective (RPO) 9
- Recovery point objective (RPO) 30, 41
- Recovery Time Objective (RTO) 9
- Recovery time objective (RTO) 30, 41
- Redbooks Web site 234

- Contact us xiii
- Replication
 - application level 27
 - database level 26
 - disk level 26
- Replication, of information 26
- Resource Access Control Facility (RACF) 57
- Resource Access Control Facility (RACF®) 57
- Resource capping 65
- Resource Measurement Facility (RMF) 117
- RMODE 60

S

- Scalability 82
- SECLABEL 88
- secondary router (SECRouter) 69
- Secure Socket Layer (SSL) 90
- Security
 - z/VM
 - Lighthouse Directory Access Protocol (LDAP) 87
 - Resource Access Control Facility (RACF) 87
- Security label 88
- Service Level Agreement (SLA) 9
- Service Level Objectives (SLOs) 119
- Service Request Block (SRB) 61
- service virtual machine (SVM) 66
- Single Point of Failure (SPOF) 76
- Single-Sign On 162
- Specialty engine 53
- Start Interpretive Execution (SIE) 88
- static VIPA 70
- Storage Area Network (SAN) 81
- storage-protect keys 86
- Structured Query Language (SQL) 159
- Sysplex Distributor 69
- Sysplex Distributor (SD) 71
- System Assist Processor (SAP) 124
- System availability 3
- System Data Mover (SDM) 80
- system managed storage (SMS) 122
- System Management Facility (SMF) 117
- System Modification Program Extended (SMP/E) 121
- System z Application Assist Processor (zAAP) 124
- System z Integrated information Processor (zIIP) 125
 - eligible workload 127
- Systems management
 - asset and financial management 124
 - asset management 124
 - availability and performance management 114
 - availability management 114
 - change management 120
 - configuration management 121
 - financial management 124
 - incident management 123
 - performance management 115
 - problem management 119
 - real time monitors, at infrastructure level 117
 - real time monitors, at subsystems level 117
 - release management 121
 - security management 118

- service level management 119
- storage management 122
- system automation 116
- workload scheduler 116

T

- Task Control Block (TCB) 61
- Tier, in application 148
- Tivoli Configuration Manager 121
- Tivoli Enterprise Console (TEC) 118
- Tivoli Enterprise Portal (TEP) 117
- Tivoli Monitoring for Linux on System z agent 117
- Tivoli NetView for z/OS 117
- Tivoli Omegamon XE for CICS on z/OS 117
- Tivoli Omegamon XE for CICSTG on z/OS 117
- Tivoli Omegamon XE for DB2 Performance Monitor (PM) 118
- Tivoli Omegamon XE for IMS on z/OS 117
- Tivoli Omegamon XE for Mainframe Networks 117
- Tivoli Omegamon XE for Messaging 118
- Tivoli Omegamon XE for Storage on z/OS 118
- Tivoli Omegamon XE on z/OS 117
- Tivoli Omegamon XE on z/VM and Linux 117
- Tivoli Omegamon for DB2 Performance Expert (PE) 118
- Tivoli Service Level Advisor (TSLA) 119
- Tivoli Service Request Manager 121
- Tivoli Usage and Accounting Manager (TUAM) 124
- TotalStorage Productivity Center for Replication (TPC-R) 134
- Transformation matrix 148
- Type 1 hypervisor 53

U

- Unavailability 11
- Uninterruptable Power Supply (UPS) 12
- UNIX System Services 106

V

- vertical scalability 82
- VIPA (Virtual IP Addressing) 70
- Virtual cluster 20
- Virtual IP Addressing (VIPA) 70
- Virtual LAN (VLAN) 72
- virtual machine (VM) 55
- Virtual Machines Resource Manager (VMRM) 66
- virtual memory 50
- virtual storage 50
- virtualization 52
- VLAN tagging 69
- VM (virtual machine) 55
- VMRM-CMM (VMRM Cooperative Memory Management) 66
- VMRM-CMMA (VMRM Collaborative Memory Management Assist) 66

W

- Workload Manager (WLM) 50
- Write-ahead log (WAL) 105

X

X.25 68

X/Open XA 159

Z

z/OS 50

z/OS Distributed File Service zSeries File System (zFS)
106

z/OS Global Mirror (XRC) 80

z/VM 51

Control Program (CP) 57

guests 51

IEEE 802.1Q standard 73

TCP/IP 56

TCP/IP stack 72

virtual machine 51

Virtual networking 71

Virtual Switch 73

virtualization 57

VLAN support 72

zAAP on zIIP 127

Considerations for Transitioning Highly Available Applications to System z

(0.2"spine)
0.17"<->0.473"
90<->249 pages



Redbooks®

Considerations for Transitioning Highly Available Applications to System z

Understanding your workload

You may have several triggers to investigate the feasibility of moving a workload or set of workloads to the IBM System z platform. These triggers could be concerns about operational cost, manageability, or delivering the agreed service levels, among others.

Choosing between technology options on System z

Investigating the feasibility of a possible migration or transition to any other platform, including System z, requires a number of basic steps. These steps usually start with an understanding of the current workload and its pain points, and end with a business case to move the workload. It is important to find out how easy a migration is going to be and how much risk will be involved.

Making transition decisions

In this IBM Redbooks publication we offer thoughts on how to move through these steps. We also include a chapter with a System z technology summary to help you understand how a migrated workload may fit on the platform.

Our focus in this book is on workloads that are mission-critical and require a high level of availability, including disaster recovery.

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

SG24-7824-00

ISBN 0738435856