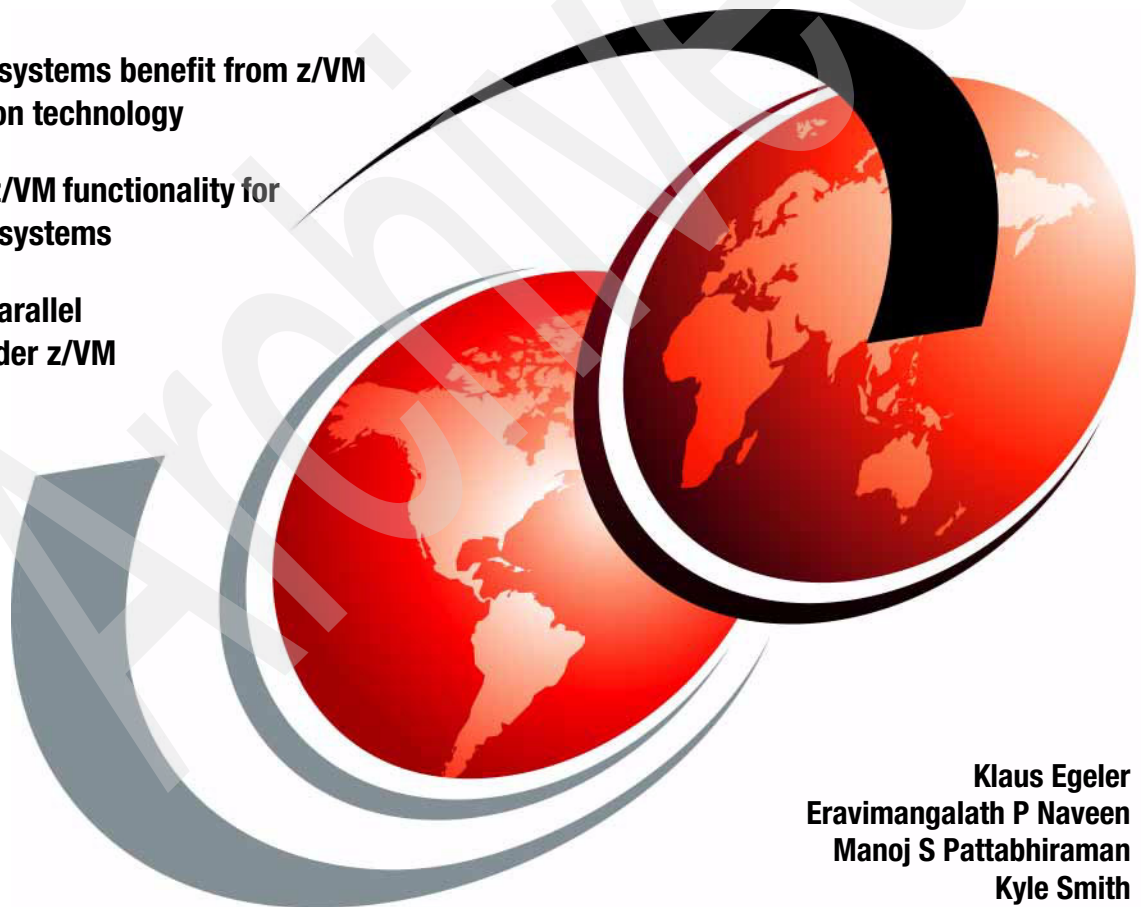


Using z/VM for Test and Development Environments: A Roundup

How guest systems benefit from z/VM virtualization technology

Exploiting z/VM functionality for your guest systems

Testing a Parallel Sysplex under z/VM



Klaus Egeler
Eravimangalath P Naveen
Manoj S Pattabhiraman
Kyle Smith



International Technical Support Organization

**Using z/VM for Test and Development
Environments: A Roundup**

February 2007

Archived

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

First Edition (February 2007)

This edition applies to z/VM Version 5, Release 2 (product number 5741-A05).

© Copyright International Business Machines Corporation 2007. 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 that wrote this IBM Redbook	xi
Become a published author	xiii
Comments welcome	xiii
Part 1. Introducing z/VM	1
Chapter 1. z/VM virtualization	3
1.1 General overview of z/VM	4
1.2 z/VM virtualization technology provides guest support	5
1.3 z/VM provides proven system integrity, security, and reliability	6
1.4 A brief history of z/VM	7
1.5 Running guest operating systems	8
1.5.1 Guest support	8
1.5.2 System simulation	9
1.5.3 System efficiency	9
1.6 Benefits of using z/VM in a test, development, and production environment	12
1.6.1 Running Linux as guest system	13
1.6.2 Running TPF as guest system	15
1.6.3 Running z/OS as a guest system	15
1.6.4 Running z/VSE as guest system	19
1.6.5 Running z/VM as guest system	20
Chapter 2. Virtual networking for Linux on System z	21
2.1 Network virtualization	22
2.2 Guest LAN	22
2.2.1 Steps involved in setting up a guest LAN	24
2.2.2 Benefits of the guest LAN in an integrated environment	26
2.3 VSWITCH	27
2.3.1 VSWITCH configurations	30
2.3.2 Configuring a Layer 2 VSWITCH in z/VM	32
2.3.3 Benefits of VSWITCH on an integrated environment	33
2.4 HyperSockets Network Concentrator	33
2.4.1 Setting up concentrator	35
2.4.2 Setting up leaf nodes	35

2.4.3	Starting the concentrator	36
2.4.4	Stopping the Concentrator	36
2.5	High availability	37
2.5.1	High availability with z/VM and Linux on System z	37
2.6	Channel bonding	38
2.6.1	Configuring bonding devices	39
2.6.2	Benefits of channel bonding	42
2.7	VIPA with dynamic routing	42
2.7.1	Dynamic routing	44
2.7.2	VIPA configuration	45
2.7.3	Dynamic routing configuration	46
2.7.4	Starting dynamic routing daemons	47
2.7.5	Verifying the router	47
Chapter 3. Resource management under z/VM		49
3.1	Importance of resource management under z/VM	50
3.2	Resources that can be changed dynamically under z/VM	50
3.3	Managing dynamic DASD changes under z/VM	51
3.3.1	Dynamically adding a DASD device to a z/VM virtual machine	51
3.3.2	Dynamically removing a DASD device from a z/VM virtual machine	53
3.3.3	Dynamically adding a DASD device to a z/VM Linux guest	54
3.3.4	Dynamically removing a DASD device from a z/VM Linux guest	54
3.4	Managing dynamic network changes under z/VM	55
3.4.1	Dynamically adding a NIC to a z/VM virtual machine	56
3.4.2	Dynamically removing a NIC from a z/VM virtual machine	57
3.4.3	Dynamically adding a NIC to a z/VM Linux guest	58
3.4.4	Dynamically removing a NIC from a z/VM Linux guest	60
3.5	Managing dynamic CPU changes under z/VM	60
3.5.1	Dynamically adding a CPU to a z/VM virtual machine	61
3.5.2	Dynamically removing a CPU from a z/VM virtual machine	62
3.5.3	Dynamically adding a CPU to a Linux guest running on z/VM	62
3.5.4	Dynamically removing a CPU from a z/VM Linux guest	64
3.6	Storage management under z/VM	66
3.7	Resource simulation using z/VM	67
3.7.1	Simulating a multiprocessor system using z/VM	68
3.7.2	Simulating an FBA DASD device using z/VM	70
3.8	References	72
Chapter 4. Outlook on z/VM Version 5 Release 3.0		73
4.1	Improvements in z/VM 5.3.	74
4.1.1	General outlook	74
4.1.2	Processor and device support	74
4.1.3	Networking	75

4.1.4 Virtualization	75
4.1.5 Systems management	75
4.1.6 Security	76
4.1.7 Guest ASCII console support	77
Part 2. Developing, deploying, and managing your environment	79
Chapter 5. Cloning a production environment (a practical scenario) . . .	81
5.1 Introducing the sample environment	82
5.2 Best practices for replicating environments.	82
5.2.1 Determining storage and paging allocation.	83
5.2.2 Allocating DASD	83
5.2.3 Managing the CP directory	85
5.3 Planning considerations	85
5.3.1 Determining device usage in Linux	85
5.3.2 Determining device usage in z/VM	86
5.3.3 Determining device usage in z/OS	87
5.4 Setting up z/VM to mimic your existing environment.	88
5.4.1 Creating a virtual network	88
5.4.2 Adding guests to the CP directory.	88
5.4.3 Copying DASD volumes	90
5.4.4 Updating settings for cloned systems	92
5.4.5 Testing the changes	95
Chapter 6. Parallel Sysplex under z/VM	97
6.1 Why Parallel Sysplex under z/VM?	98
6.2 Parallel Sysplex under z/VM: Planning and steps.	99
6.2.1 Planning to set up a Parallel Sysplex under z/VM	99
6.2.2 Steps to set up a Parallel Sysplex under z/VM	99
6.2.3 Establishing data sharing among the z/OS virtual machines in a virtual Parallel Sysplex	100
6.3 Procedure to create a CF failure under z/VM	102
6.4 Procedure to recover a CF under z/VM.	104
6.5 Procedure to create a CF connectivity failure under z/VM	105
6.6 Procedure to re-establish the CF connectivity under z/VM	107
6.7 Failing and recovering a system with z/VM	109
6.8 References	110
Chapter 7. z/TPF	111
7.1 History and general overview of z/TPF	112
7.1.1 History of TPF	112
7.1.2 Overview of TPF	113
7.1.3 TPF application compilation with z/OS	115
7.2 Running z/TPF as guest under z/VM	115

7.2.1	Benefits of running z/TPF as guest under z/VM	115
7.2.2	z/TPF application compilation with Linux under z/VM	116
7.3	Some more information about the TPF-gcc cross compiler	117
7.4	Obtaining and using TPF-gcc	118
7.5	Post-installation steps	119
7.5.1	Verifying the installation	119
Chapter 8.	The Communication Controller for Linux on System z	121
8.1	What is the Communication Controller for Linux?	122
8.2	Useful CCL resources	122
Chapter 9.	z/VM system management tools	123
9.1	System management tools	124
9.2	IBM system management tools	125
9.2.1	IBM Virtualization Engine	125
9.2.2	IBM Director	126
9.3	z/VM Center	127
9.3.1	Server complexes	128
9.4	Benefits of IBM Director and z/VM Center	129
9.5	IBM Tivoli Provisioning Manager	130
9.5.1	Tivoli Provisioning Manager components	130
9.5.2	Benefits of IBM Tivoli Provisioning Manager	132
9.6	Aduva OnStage	132
9.7	Performance toolkit	133
9.7.1	Performance toolkit functions	133
9.7.2	Modes of operations	134
9.7.3	Linux monitoring with performance toolkit	134
9.8	Hobbit monitor	135
9.8.1	Features of Hobbit monitor	135
9.8.2	z/VM client extension to Hobbit monitor	136
Chapter 10.	Porting applications to Linux on System z	137
10.1	Application portability	138
10.1.1	Linux on System z porting considerations	138
10.2	Migration Toolkit for Linux on System z	139
10.2.1	Components of Migration Toolkit	140
10.2.2	GUI Front-end	140
10.2.3	Code Scanner	140
10.2.4	Benefits of Migration Toolkit for Linux	141
Part 3.	Appendixes	143
Appendix A.	Sample scripts to use when cloning systems	145
	Copying production volumes	146

Updating Linux network settings	148
Updating WebSphere Portal Server instance data	150
Appendix B. Sample REXX script to IPL z/OS systems on z/VM	153
IPL EXEC	154
Appendix C. Additional material	157
Locating the Web material	157
Using the Web material	158
System requirements for downloading the Web material	158
How to use the Web material	158
Related publications	159
IBM Redbooks	159
Other publications	159
Online resources	160
How to get IBM Redbooks	162
Help from IBM	162
Index	163

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

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

AIX®	NetView®	System z™
DB2®	OS/390®	SystemView®
DFSMS/VM™	Parallel Sysplex®	Tivoli®
DirMaint™	PartnerWorld®	TotalStorage®
DS8000™	Power PC®	Virtualization Engine™
ECKD™	RACF®	VM/ESA®
Enterprise Systems	Redbooks (logo)  ™	VSE/ESA™
Architecture/390®	Redbooks™	VTAM®
ESCON®	REXX™	WebSphere®
eServer™	RMF™	z/Architecture®
FICON®	S/360™	z/OS®
FlashCopy®	S/390®	z/VM®
HiperSockets™	System i™	z/VSE™
IBM®	System p™	z9™
Language Environment®	System x™	zSeries®
MVS™	System z9™	

The following terms are trademarks of other companies:

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

NOW, and the Network Appliance logo are trademarks or registered trademarks of Network Appliance, Inc. in the U.S. and other countries.

IPX, Java, Solaris, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

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

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

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

Preface

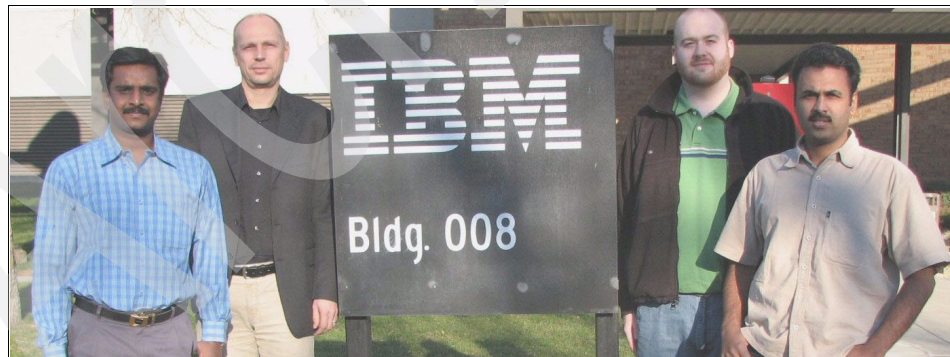
This IBM® Redbook shows some of the strengths of IBM z/VM® and how you can use these strengths to create a highly flexible test and development environment.

One IBM System z™ server running z/VM Version 5 (z/VM V5) can do the job of many distributed servers scattered across the enterprise by hosting a variety of IBM operating platforms such as Linux® on IBM System z, z/OS®, z/OS.e, z/VM, Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA™), z/VSE™, Transaction Processing Facility (TPF), z/TPF, and z/VM itself.

Additionally, this book discusses how z/VM enables dynamic resource management. Resource management under z/VM is very important because z/VM uses real resources assigned to it to create and control *virtual machines*, control real devices, and simulate devices for use by virtual machines.

The team that wrote this IBM Redbook

This IBM Redbook was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Poughkeepsie Center.



The team (from left to right): Manoj S Pattabhiraman, Klaus Egeler, Kyle Smith, and Eravimangalath P Naveen

Klaus Egeler is an IT Systems Management Specialist with IBM Global Services, Germany. He has more than 15 years of experience as a VSE and VM systems programmer. He has worked with Linux for IBM eServer™ zSeries®

and IBM S/390® for more than five years. He has contributed to several z/VM and Linux related IBM Redbooks™ and was a speaker at ITSO workshops and customer events.

Eravimangalath P Naveen is an IT Infrastructure Architect at the IBM India Software Lab in Bangalore, India with seven years of experience. His areas of expertise include IBM AIX®, Linux, z/VM, IBM Tivoli® Storage Manager, VMWare, Storage systems, Networking, and Virtualization across all IBM server systems.

Manoj S Pattabhiraman is a staff software engineer in Linux Technology Center (LTC), India. He has seven years of experience as a z/VM Application developer and in Systems programming for Linux on zSeries. He holds a Masters degree in Computer Applications. His areas of expertise include Linux on zSeries, Middleware performance and z/VM REXX™ programming.

Kyle Smith is a software engineer at the IBM Test and Integration Center for Linux in Poughkeepsie, New York, where he tests IBM middleware and enterprise Linux distributions on IBM System z. He holds a Bachelor of Science degree in computer science from Clarkson University. His areas of expertise include Linux and Java™ application development.

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

Lydia Parziale, Project Leader
ITSO, Poughkeepsie Center

Robert J. Brenneman
Philip F. Chan
Sujata P. Nanda
Jin Xiong
IBM Test and Integration Center for Linux, Poughkeepsie

Rich Conway
ITSO, Poughkeepsie Center

Roy P Costa
ITSO, Poughkeepsie Center

Samuel D. Cohen
IBM Global Technology Services, Phoenix

Alan C. Altmark
IBM Systems and Technology Group, Endicott

Bruce J. Hayden
IBM Integrated Technology Delivery, Endicott

Become a published author

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

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

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

ibm.com/redbooks

- Send your comments in an e-mail to:

redbooks@us.ibm.com

- Mail your comments to:

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



Part 1

Introducing z/VM

z/VM provides a highly flexible test and production environment for enterprises deploying on demand business solutions. z/VM exploits the IBM z/Architecture® and helps enterprises meet their growing demands for multi-user server solutions with a broad range of support for operating platforms such as Linux on IBM System z, z/OS, z/OS.e, VSE/ESA, z/VSE, Transaction Processing Facility (TPF), and z/TPF. You can refer to the following Web site:

<http://www.vm.ibm.com/index.html>

In this part, we provide an explanation of what z/VM is, what features and components are available in Version 5 Release 2, and give an overview of some of the networking options available.

z/VM virtualization

In this chapter, we give a basic overview of z/VM. We describe how z/VM uses virtualization and discuss the advantages of using z/VM in a test and development environment as a hypervisor for guest operating systems.

In this chapter, we discuss the following topics:

- ▶ A general overview of z/VM
- ▶ A brief history of VM
- ▶ Running guest operating systems
- ▶ The benefits of using z/VM in a test, development, and production environment

1.1 General overview of z/VM

Note: The contents of this chapter are related to z/VM Version 5 Release 2 unless otherwise noted.

z/VM is an IBM z/Architecture (64-bit) operating system for the IBM System z platform. z/VM provides a highly flexible test and production environment for enterprises deploying the latest on demand business solutions. The z/VM implementation of the IBM virtualization technology provides the capability to run other full-function operating systems, such as Linux and z/OS, under z/VM as *guest* systems. z/VM supports 64-bit z/Architecture guests and 31-bit IBM Enterprise Systems Architecture/390® guests.

The z/VM base product includes the following components and facilities:

- ▶ Control Program (CP)
- ▶ Conversational Monitor System (CMS)
- ▶ Transmission Control Protocol/Internet Protocol (TCP/IP) for z/VM
- ▶ Advanced Program-to-Program Communication/Virtual Machine (APPC/VM) Virtual Telecommunications Access Method (VTAM®) Support (AVS)
- ▶ Dump Viewing Facility
- ▶ Group Control System (GCS)
- ▶ Hardware Configuration Definition (HCD) and Hardware Configuration Manager (HCM) for z/VM
- ▶ IBM Language Environment®
- ▶ Open Systems Adapter Support Facility (OSA/SF)
- ▶ Restructured Extended Executor/Virtual Machine (REXX/VM)
- ▶ Transparent Services Access Facility (TSAF)
- ▶ Virtual Machine Serviceability Enhancements Staged/Extended (VMSES/E)

z/VM also offers the following optional features:

- ▶ IBM Data Facility Storage Management Subsystem for VM (DFSMS/VM™)

Note: DFSMS/VM is an optional feature of the z/VM System Delivery Offering (SDO).

- ▶ IBM Directory Maintenance Facility (DirMaint™)
- ▶ Performance Toolkit for VM
- ▶ IBM Resource Access Control Facility (RACF®) for z/VM

1.2 z/VM virtualization technology provides guest support

z/VM presents a unique approach to computer operating systems. It 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, storage, 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.

The virtual machine capability of z/VM allows you to:

- ▶ Test programs that can cause abnormal termination of real machine operations and, at the same time, process production work. The isolation that is provided for a virtual machine enables system-oriented programs and teleprocessing applications, for example, to be tested on the virtual machine while production work is in progress, because this testing cannot cause abnormal termination of the real machine.
- ▶ Test a new operating system release. A new release of an operating system can be generated and tested at the same time that the existing release is performing production work. This enables the new release to be installed and put into production more quickly. The ability to operate multiple operating systems concurrently under z/VM may enable an installation to continue running programs that operate only under a back-level release (programs that are release-sensitive and uneconomical to convert, for example) concurrently with the most current release.
- ▶ Test a new operating system. The existing operating system can be used to process production work concurrently with the generation and testing of a new operating system. Experience with the new system can be obtained before it is used on a production basis, without dedicating the real machine to this function.

- ▶ Perform operating system maintenance concurrently with production work. The installation and testing of program temporary fixes (PTFs) for an operating system can be done at the same time normal production operations are in progress.
- ▶ Provide backup facilities for the primary operating system. A generated z/VM system is not model-dependent and can operate on various server models as long as the minimum hardware requirements are present. This enables a smaller server model that has less real storage, fewer channels, fewer direct access devices, and fewer unit record devices than a larger server model to provide backup for the larger model (normally at a reduced level of performance).
- ▶ Perform operator training concurrently with production work processing. The real machine does not have to be dedicated to training additional or new operators or to providing initial training when a new operating system is installed. Operator errors cannot cause termination of real machine operations.
- ▶ Simulate new system configurations before the installation of additional channels and I/O devices. The relative load on channels and I/O devices can be determined using the simulated I/O configuration rather than the real I/O configuration. Experience with generating and operating an I/O configuration for multiple guest operating systems can be obtained using one real machine.
- ▶ Test customer-written system exits. Customer-written system exits can be tested without disrupting production work.

1.3 z/VM provides proven system integrity, security, and reliability

z/VM is built on a foundation of system integrity and security, and incorporates many design features for reliability and availability.

- ▶ Integrity and security:
 - z/VM supports guest use of the cryptographic facilities provided by supported IBM servers.
 - IBM will correct any integrity exposures introduced by unauthorized programs into the system.
 - Kerberos authentication and Secure Sockets Layer (SSL) support are provided through TCP/IP for z/VM.
 - Integrated access control and authentication services can be augmented with the addition of the RACF for z/VM feature or other external security managers.

- ▶ Availability and reliability:
 - Application recovery: z/VM provides services that permit recovery of incomplete interactions with resource managers.
 - Automated operations: z/VM offers several levels of automated system management support. One example is the Programmable Operator. For a higher degree of automation, IBM SystemView® Host Management Facilities/VM can be added. Both the Programmable Operator and Host Management Facilities/VM can interact with IBM NetView® on z/VM, which in turn can interact with NetView on z/OS.
 - z/VM provides duplexed data with transparent ongoing synchronization between the primary and backup copy, and automatic transparent switching to the backup copy in case of an error in the primary copy.
 - Online configuration changes eliminate many previously required outages.
 - z/VM systems can be connected for improved server and user availability.
 - Fast restart reduces the user impact of any outage.

1.4 A brief history of z/VM

The current version of z/VM at the time of writing this book is z/VM Version 5 Release 2. VM was first introduced as VM/370 in 1972 and was enhanced and developed ever since. Figure 1-1 shows the evolution of z/VM starting with z/VM Version 3 Release 1, which was in general availability (GA) in February 2001. In Figure 1-1, you can see the status of the different z/VM versions and the date when it went into that status. For example, z/VM Version 3 Release 1 was withdrawn from marketing and service and was discontinued in December 2005.

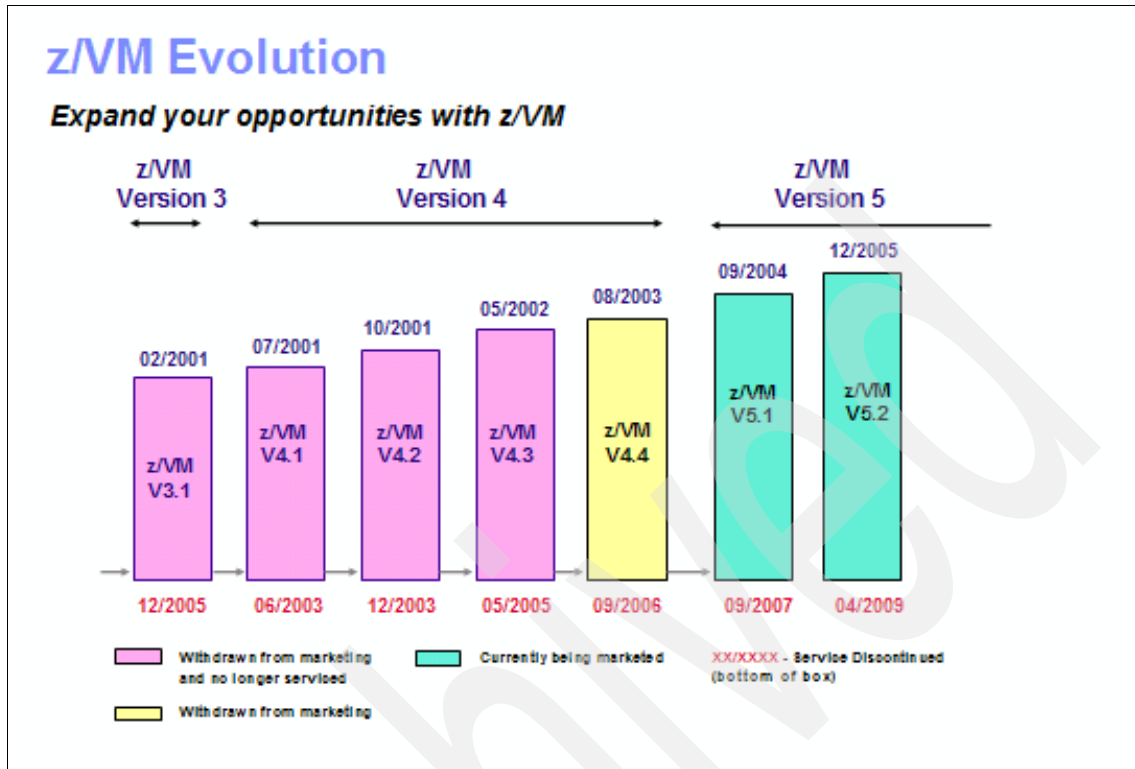


Figure 1-1 The evolution of z/VM

1.5 Running guest operating systems

There are some advantages of using guest support in z/VM.

1.5.1 Guest support

You can use guest support in z/VM to develop, maintain, manage, and migrate other operating systems that make use of IBM Enterprise Systems Architecture/390 (ESA/390) or IBM z/Architecture. In fact, these tasks constituted the original VM mission. When performing these tasks, system programmers and application programmers often solved the problems they encountered by using the solutions that were already integrated or implicit in z/VM. In time, z/VM developed a large collection of handy, useful tools. So much so that today many IBM customers choose to run their systems in virtual machines as guests of z/VM. If you ask them why, they offer an impressive list of reasons.

1.5.2 System simulation

Sooner or later, every computer user wants a spare system to try out an idea or to isolate a task. z/VM is capable of making available almost limitless *spare systems* in virtual machines, each simulating a real machine with very high fidelity. Some of the advantages of making one system look like many are:

- ▶ You can create an exact replica of your production system on which you can test your new programs, services, and procedures. This is a relatively inexpensive way to have your own test system (or as many test systems as you would like). It is also a safe way to test a new function, because your real production system, its applications and data, are protected from any damage that the new function might cause if you tested it on the host system.
- ▶ You can have a flexible migration system. This eliminates the usual inconvenience to users which migration usually causes.
- ▶ You can create a multiple production system environment. For example, your installation might run VSE applications, be migrating to z/OS, and have a new vendor package that runs under Linux. As guests of z/VM, all three can run efficiently while sharing one processor.

1.5.3 System efficiency

z/VM's guest support helps you to run systems at lower cost but with greater efficiency. Some of the benefits are discussed in the following sections.

Performance

Guest systems may see performance improvements by exploiting z/VM features. For example, both virtual disk in storage and minidisk cache allow guests to avoid real I/O by using data in storage and caching techniques.

Reduced hardware costs

Guest systems can share devices such as channels, printers, and direct access storage device (DASD), which z/VM efficiently manages. Indeed, z/VM adds new value to such devices merely by the way it manages them. A good example is z/VM's minidisk support, which allows one real disk to function as though it were several smaller disks (such as multiple IPLable minidisks).

Due to the fact that z/VM simulates some devices (such as unit record devices and channel-to-channel (CTC) adapters), your installation might be able to avoid buying real ones.

Operations management

With PROP (z/VM's PProgrammable OPerator), you can cut your console traffic substantially. That saves time and reduces errors.

Recovery management

You can build guest virtual machines to simulate systems at your organization's other sites. Therefore, z/VM can become a disaster-recovery backup site.

z/VM Version 5 extends the strength of the IBM System z environment as an ideal platform for integrating applications and data and consolidating select UNIX® and Linux workloads on a single System z server. This can allow you to run tens to hundreds of Linux servers with the potential to support new workloads, achieve productivity gains, and lower your information technology (IT) costs. z/VM supports Linux on the mainframe and enables Linux servers to share hardware resources and use internal high-speed communications within the System z environment.

z/VM Version 5 allows you to put the power of System z server partitioning and z/VM virtualization technology to work for you to help realize the benefits of workload isolation and resource sharing, including:

- ▶ Reliability, availability, and serviceability of IBM mainframes
- ▶ Flexibility to support as many as 60 logical partitions (LPARs) on the IBM System z9™ Enterprise Class (formerly the IBM System z9 109)
- ▶ Ability to virtualize each LPAR into hundreds or more virtual machines
- ▶ Ability to virtualize processor, communication, memory, storage, I/O, and networking resources
- ▶ Help with maximizing resources to achieve high system utilization
- ▶ Advanced dynamic resource allocation
- ▶ High-speed communications among LPARs and guests with IBM HiperSockets™
- ▶ Advanced systems management, administration, and accounting tools

The success of Linux on System z can be largely attributed to the business value that Linux-based solutions derive from the IBM mainframe virtualization technology provided by z/VM and System z servers. z/VM technology offers you the ability to host a large number of Linux servers on a single mainframe while also providing an operational environment that is well-suited for on demand computing: Highly flexible, adaptable, and efficient.

z/VM Version 5 offers new levels of price/performance, functional capabilities, and hardware exploitation that are expected to increase the attractiveness of

deploying Linux solutions on the mainframe. You can add capacity to IBM mainframe servers for hosting Linux on z/VM workloads by configuring your servers with Integrated Facility for Linux (IFL) engines.

The z/VM Version 5 pricing model can make it more feasible for you to add z/VM virtualization technology to your standard-engine environment (compared with the pricing models for z/VM Version 3 and Version 4). This enables you to use z/VM to host workloads that cannot run on IFL engines, such as z/OS, z/VSE, VSE/ESA, z/TPF, or TPF, to consolidate existing IBM VM/ESA® or z/VM Version 3 and Version 4 workloads onto a single, larger System z server.

z/VM Version 5 is the successor product to VM/ESA, z/VM Version 3, and z/VM Version 4. z/VM Version 5 provides additional support and exploitation opportunities for the thousands of users who have built enterprise-wide automation and infrastructure enhancements on the VM platform in support of their applications, database systems, and on demand business solutions.

z/VM Version 5 is intended to address the following situations:

- ▶ Running more Linux server images on a single physical server.

Considerably more images than are currently supported by the LPAR mode of operation can be achieved with z/VM guest support. These Linux on System z server images can be deployed on standard central processor engines (CPs) or IFL engines with z/VM Version 5. Running multiple Linux images on an IFL-configured z/VM system may not increase the IBM software charges of your existing System z environment. You can add z/VM Version 5 running on IFL engines to your existing z/OS, z/OS.e, z/VM, z/TPF, TPF, z/VSE, VSE/ESA, or Linux on System z environments without increasing software costs on the standard engines (CPs).

- ▶ Moving select Linux, Windows®, and UNIX workloads to a single physical server while maintaining distinct server images and current LAN topology.

This ability can help reduce systems management complexity. Because the number of real hardware servers and associated physical LANs is reduced, cost savings can be realized by managing large server farms deployed on virtual servers instead of using multiple hardware servers. Deploying Linux workloads on z/VM Version 5 is particularly attractive if they interact with applications or data located on the same System z server.

- ▶ Enhancing virtual networking.

z/VM virtual switch support provides external connectivity for guest LANs through an OSA-Express adapter without requiring a z/VM or Linux router virtual machine.

- ▶ Consolidating operating systems on the System z platform.

z/VM V5.2 provides considerable relief for real-memory-constrained z/VM environments. The z/VM Control Program provides added support for large real-memory configurations, improving the effectiveness of storage located above the 2 GB address bar. This might enable a single z/VM instance to support a significantly larger number of guest operating systems, or guests with greater aggregate memory, than prior z/VM releases. This constraint relief is provided for both z/Architecture and ESA/390 guest operating systems, such as Linux for System z, z/TPF, TPF, z/OS, z/OS.e, z/VSE, VSE/ESA, or z/VM. If you are experiencing real-memory constraints below the 2 GB address line with z/VM V5.1 and earlier levels of z/VM and VM/ESA, you are likely to experience improved system throughput with z/VM V5.2.
- ▶ Migrating from VM/ESA to z/VM Version 5. This helps to enable:
 - More memory to cache CMS minidisks by exploiting memory above 2 GB for minidisk cache.
 - Connectivity (TCP/IP) enhancements and additional device support.
 - Added security with SSL-protected TCP/IP network traffic, such as Telnet sessions and Web transactions.
- ▶ Migrating to the new version or release of a guest operating system using z/VM Version 5.

This can provide added flexibility for migration, production, and testing. For example, z/VM can help you migrate from IBM OS/390® to z/OS or z/OS.e.
- ▶ Enhancing guest IBM Parallel Sysplex® support in z/VM with the exploitation of z/Architecture.

This enables addressability of larger amounts of real and virtual memory, allowing the development and testing of 64-bit Parallel Sysplex applications in a guest environment.

Tip: See Chapter 6, “Parallel Sysplex under z/VM” on page 97, for additional information.

1.6 Benefits of using z/VM in a test, development, and production environment

To summarize the benefits of using z/VM, we show the specific advantages of using z/VM as a hypervisor for each guest operating system.

1.6.1 Running Linux as guest system

This section describes some of the advantages of running Linux as a guest system of z/VM.

z/VM functions that Linux can exploit

Some specific z/VM product functions that can be exploited by Linux include performance, Reliability, Availability and Serviceability (RAS), operation and management, and productivity.

► **Performance**

- VM provides multi-image support and can run hundreds of Linux guests simultaneously.
- Using IUCV or virtual channel-to-channel adapter (CTCA) devices, VM supports very high-performance networking among Linux guests.
- These same facilities can be used to establish high-speed connections to the VM TCP/IP stack.
- VM exploits large N-way processor architectures that Linux cannot yet use effectively. Coupled with the ability of VM to run multiple Linux guests, this allows more resources to be devoted to Linux-based applications on a single system.
- Data-in-memory support, provided by VM virtual disks in storage and minidisk caching, provides transparent, high-speed data access for Linux guests.
- VM exploits expanded storage on behalf of Linux guests automatically and transparently.

► **RAS**

- Minidisk I/O and processor error recovery are built into VM and provided automatically, relieving the Linux guest of this responsibility.
- Dynamic I/O reconfiguration support in VM allows real devices to be added to or removed from the configuration without disturbing Linux guests.
- Support for CP sparing in VM allows additional processor capacity to be brought online as needed.
- VM simplifies the ability to provide standby systems for immediate backup of failing applications.

► **Operation and management**

- VM tracing, diagnostic, and debugging facilities are extensive, simplifying kernel and device driver development, as well as supporting problem data capture and analysis for production systems.
- Extensive performance measurement, reporting, and control facilities in VM can be used to manage Linux guests.
- Facilities for virtual machine scheduling and automation can be extended to perform these functions for Linux guests.
- Because no real resources have to be dedicated to a Linux virtual machine, creating and deleting them is quick and easy.

► **Productivity**

- The Linux minidisk driver uses VM facilities to access DASD and thus supports all devices that z/VM and VM/ESA support in a device-independent manner and with complete error recovery.
- Temporary disks can be used to meet interim needs for additional Linux DASD space.
- VM simulates and virtualizes a variety of devices that Linux can exploit, including CTCA and 3215 console devices.
- VM can be used to create a complex environment for test purposes without requiring real hardware duplication.
- A Linux virtual machine may define and use up to 64 virtual processors, regardless of the number of real processors in the hardware configuration.
- VM facilitates resource sharing of CPUs, memory, DASD, and network that can maximize resource utilization while minimizing expense.
- As need arises, additional Linux guests can be created quickly and easily to ensure that user productivity is not constrained.
- The Linux kernel can be saved as an IPLable named system to simplify the boot process and ensure that one or more standard kernel configurations are available to all Linux guests.

Tip: Read the “Ten great reasons to run Linux as a guest of z/VM” at the following Web page:

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

For more information, also see the following Web page:

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

1.6.2 Running TPF as guest system

When running the System z Transaction Processing Facility (z/TPF) as a guest under z/VM, you can take advantage of using a Linux guest, running under the same z/VM hypervisor, to compile the z/TPF application.

In TPF and its predecessors, the build environment was IBM MVS™ (later OS/390 now z/OS). This meant that TPF shared the compiler, language libraries, and programming model with z/OS. It also meant that development tooling was confined to what worked with z/OS and be customized for the target system approach.

z/TPF is significantly different from the prior version of TPF in that the build environment is Linux, not z/OS. This can be any Linux. Except for building the z/TPF operating system itself, the development and build can be on any Linux server on any platform (Intel® or IBM Power PC®, for example).

z/TPF depends on the “Gnu is Not Unix”, or simply GNU, toolchain, which includes the GNU Compiler Collection (GCC) and the common open language libraries such as glibc. Because GCC is a true cross-compiler, it can run anywhere and build for the target z/TPF system.

Assembler programs are handled as IBM High Level Assembler (HLASM) (program assembler), which has been modified to produce output that matches the Linux linkage model, ELF (which z/TPF uses). The term for shared libraries in Linux is shared objects; hence, all applications (assembler, C or C++) get loaded as shared objects. This, as well as the presence of POSIX API structures in z/TPF, means that the runtime environment as well as the build environment for the application programmer is extremely similar to Linux.

Note: For more information about z/TPF, see the following Web site:

<http://publib.boulder.ibm.com/infocenter/tpfhelp/current/index.jsp>

The IBM Redbook *z/TPF and WebSphere Application Server in a Service Oriented Architecture*, SG24-7309, is also a good source to get more information.

1.6.3 Running z/OS as a guest system

Besides the ability to run z/OS systems as guest systems under z/VM, z/VM can simulate advanced *coupling facility* (CF) and *message facility* (MF) functions. Namely, the ability of a set of CFs to be directly connected to one another. z/VM Guest Coupling Simulation provides for the simulation of one or more complete Parallel Sysplexes within a single z/VM system image. The intent is to provide a

preproduction testing platform for a coupled-system installation. The z/VM simulated environment is not intended for production use because its single points of failure negate the intent of the Parallel Sysplex environment. Other than the processors required, there is no special hardware needed: No coupling links and no external coupling facilities. All guest operating systems coupled within a simulated sysplex can be coupled (through simulated coupling links) to coupling facilities also running as guests of the same z/VM system. Up to 32 virtual machines can be coupled within a simulated sysplex, with each such virtual machine coupled to up to eight coupling facility virtual machines. In addition, when the processor and z/VM are so capable, each simulated coupling facility can connect to up to seven peer simulated coupling facilities. All coupled guests and simulated coupling facilities run within a single instance of the Control Program (CP).

In order to understand the value of z/VM guest coupling simulation support, a brief description of a Parallel Sysplex environment is provided.

The purpose of a sysplex configuration is to provide a processing environment that can be available twenty-four hours a day, seven days a week, and is totally scalable. This is accomplished by connecting multiple z/OS images running on various processors or logical partitions in parallel. Although each z/OS image is independent, to the user, a single system image is reflected. Processing is balanced across all the systems within the sysplex. If any processor within the sysplex requires service or maintenance, the processor can be removed from the sysplex without affecting system availability. The other z/OS images can maintain the current workload. If more processing power is ever required, you can add new processors and additional z/OS images to the sysplex at any time.

For the multiple z/OS images to run as a single processing complex, there has to be an efficient way for all the images to share information and serialize their processing. The coupling facility and message facility were the mechanisms developed to perform just that function. The coupling facility is a logical facility with its own processors, which basically contains structured storage that is accessible to all the z/OS images. The coupling facility is attached to the processors by special fiber-optic channels (the message facility) that are used to transfer commands, data, and responses between the attached processors and the coupling facility. The coupling facility will allow z/OS to maintain such things as data, common scheduling queues, status, and locks that are shared by all the z/OS images. In addition to the shared storage, the coupling facility also provides a command-set to perform operations required to control a parallel system environment.

z/VM guest coupling simulation support is the software that simulates the hardware and software required to run a z/OS sysplex environment as second level guests under z/VM.

The z/VM guest coupling simulation support consists of three components: coupling facility service machines, coupled guests, and simulated message facility. See Figure 1-2 for an illustration of the simulated coupling facility environment on z/VM.

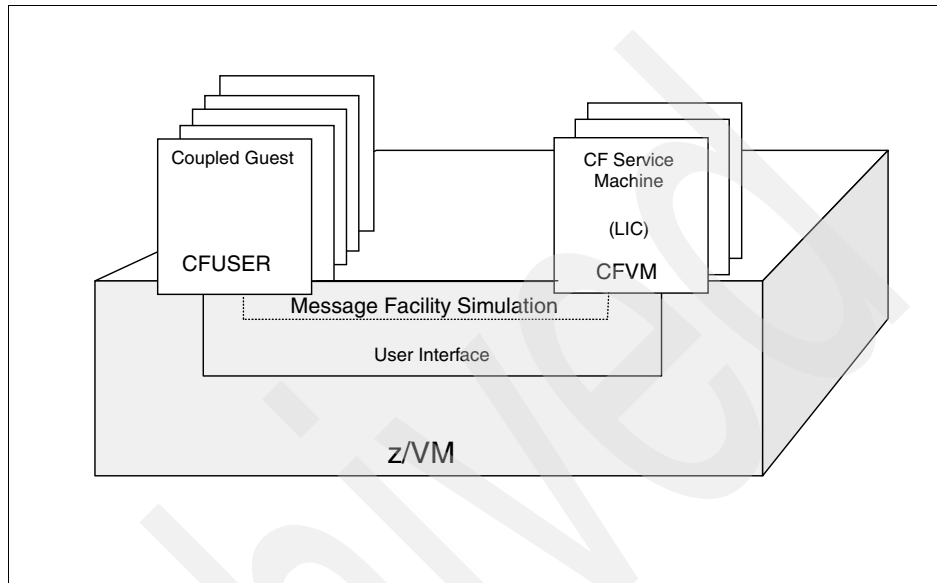


Figure 1-2 z/VM coupling facility support

The CF service machine

This is a special type of z/VM service machine that runs the coupling facility control code (CFCC) (Licensed Internal Code, or LIC). This code is not part of z/VM and is loaded directly from the processor. Because z/VM is executing the actual code that runs in a real coupling facility, there is no difference in function between a real coupling facility and z/VM's implementation.

The system administrator must define a user ID for each CF service machine that will run under z/VM. To define each user ID as a CF service machine, the CFVM operand must be specified on the OPTION directory control statement of each CF service machine. The CFVM operand defines the virtual machine as a CF service machine and allows connections between itself and coupled guests. Refer to the manual *z/VM V5R2.0 CP Planning and Administration*, SC24-6083-03, for more information about the OPTION and SPECIAL directory control statements.

Any number of CF service machines can be defined. Up to 32 coupled guests can be connected to each CF service machine. When the processor and z/VM

are enabled in this way, each CF service machine is capable of coupling up to seven other CF service machines.

The coupled guest

This is a uniprocessor (UP) or multiprocessor (MP) virtual machine that has a coupled connection to a CF service machine using the software simulation of the message facility. This virtual machine is running an operating system that supports the coupling facility, such as z/OS.

The system administrator must define a user ID for each coupled guest that will run under z/VM. To define each user ID as a coupled guest, the CFUSER operand must be specified on the OPTION directory control statement for each virtual machine. The CFUSER operand defines the virtual machine as a coupled guest and allows it to connect to a CF service machine.

Any number of coupled guests can be defined, and each coupled guest can establish a coupled connection with up to eight different CF service machines concurrently.

The message facility simulation

The message facility, which is an optional part of the channel subsystem, is the hardware that connects a processor complex to the coupling facility. CP simulates the message facility for the benefit of CF service machines and coupled guests. The simulated message facility allows a coupled guest to exchange information with a CF service machine, and when the hardware and z/VM are so capable, it allows CF service machines to exchange information with each other. From the coupled guest's point of view, the message facility appears as a set of virtual message devices. The CF service machine's view of the message facility is proprietary.

For more detailed information about z/VM Guest Coupling Simulation, see Chapter 6, "Parallel Sysplex under z/VM" on page 97, as well as *z/VM V5R2.0 Running Guest Operating Systems*, SC24-6115-01.

1.6.4 Running z/VSE as guest system

A virtual machine provides an easy, convenient way to run guest operating systems. When you run VSE or z/VSE under z/VM, you get the functional equivalent of a real processor, main and auxiliary storage, and I/O devices. When you run a guest VSE system, VSE runs under the Control Program (CP) of the z/VM system.

Advantages when running z/VSE under z/VM

The advantages of running z/VSE under z/VM are:

- ▶ More than one z/VSE system on one processor possible
- ▶ Shared I/O units possible
- ▶ Flexible communication between guest systems
- ▶ Flexible use of terminals
- ▶ Flexible configuration setup
- ▶ Advanced data base integrity
- ▶ Exploitation of expanded storage
- ▶ Broad range of CMS applications

In z/VSE, there are a set of functions to interface with the z/VSE guest from CMS, the VM/VSE Interface routines.

The VM/VSE Interface is a set of VSE phases and CMS modules supplied by z/VSE. These phases and modules provide functions for interfacing with one or more z/VSE guest systems from CMS. The VM/VSE Interface routines are distributed in IJSYSRS.SYSLIB. You must obtain the routines from the z/VSE library and install them on a CMS minidisk.

The following functions are supported by the VM/VSE Interface:

- ▶ Have none, some, or all messages from a job or from the system echoed to a specified owner (CMS user ID).
- ▶ Reply to messages resulting from the execution of a job. The job must have a unique job owner ID (CMS user ID).
- ▶ Submit jobs from a CMS terminal to a z/VSE guest system.
- ▶ Issue VSE commands (including REDISPLAY commands) to a z/VSE guest system and have the resulting messages echoed to the CMS user.
- ▶ Issue CP commands for execution in the virtual machine and have the resulting CP messages routed to the CMS job owner.

Tip: For more information about running z/VSE as guest under z/VM, see *z/VM V5R2.0 Running Guest Operating Systems*, SC24-6115-01, and *z/VSE V3R1.1 Planning*, SC33-8221-02, and for more information about the VM/VSE Interface routines, see *z/VSE V3R1.0 Operation*, SC33-8239-01.

1.6.5 Running z/VM as guest system

Running VM as a guest system, also called a second level system, of z/VM offers opportunities for more flexible system management. In this environment, you can:

- ▶ Test new application programs
- ▶ Test new releases of VM
- ▶ Test new maintenance procedures and modifications
- ▶ Train operators and system programmers

These activities can be independent of any other tasks because they are being performed in a virtual machine.

One of the biggest advantages of running a second level VM system is the ability to generate a new system without disturbing normal production activity. System programmers can log on their own virtual machines and go through the generation steps at their own pace while the rest of the installation uses the real z/VM system undisturbed. You can use XEDIT to create and update the files that are used during system generation. When the system is tested, it can be placed online, replacing the previous version with minimal disruption to the production activity.

Note: For more information about running z/VM as guest under z/VM, see *z/VM V5R2.0 Running Guest Operating Systems*, SC24-6115-01.

Virtual networking for Linux on System z

This chapter describes various virtual networking considerations that can be used to build a flexible and highly available network infrastructure on Linux for System z.

In this chapter, we discuss the following topics:

- ▶ Guest LAN
- ▶ Virtual switch (VSWITCH)
- ▶ HiperSockets network concentrator (HSNC)
- ▶ High availability examples
 - Channel bonding (layer 2 mode)
 - Virtual IP Addressing (VIPA) with Dynamic Routing (layer 3 mode)

2.1 Network virtualization

The networking options available in z/VM can be broadly split into two categories: Physical hardware and virtualization technology. Physical hardware, as the term suggests, covers physical network interfaces, or in the case of HiperSockets, a networking implementation that requires zSeries hardware. Virtualization technology covers the networking options available to those users who run Linux on a z/VM environment.

In a virtual environment, the Linux operating system runs as a guest in the virtual machine. These interfaces are a VM Control Program (CP) simulation of a physical interface.

2.2 Guest LAN

In z/VM 4.2, the CP was enhanced to provide a feature known as *guest LAN*. A guest LAN provides connectivity for a group of virtual machines within a single z/VM image allowing communication among themselves, independent of other groups of virtual machines on other guest LANs.

A guest LAN:

- ▶ Functions as a network hub connecting the z/VM guests
- ▶ Facilitates IP layer data transfers between z/VM guests
- ▶ Is a z/VM system resource, not a device

As seen in Figure 2-1 on page 23, the TCP/IP service machine owns a dedicated connection to the external network via an open systems adapter (OSA) card. The TCP/IP service machine also has a connection to the guest LAN and serves as a router to the other Linux guest machines connected to the LAN.

To participate in a guest LAN, a single virtual network interface card (NIC) must be defined for the guest. This significantly reduces the configuration effort and storage requirements of a guest machine serving as a router. Each adapter defined contains multiple OSA type devices that can be connected to a guest LAN. Guest LANs can be created and destroyed dynamically and access to a guest LAN can be restricted for security or traffic segregation requirements.

There is no architectural limit on the number of guest LAN segments that can be created. The actual limit is governed by available machine resources, but this is more of a theoretical limit than a practical one. The number of guest LANs that can be created can be considered unlimited for all practical purposes.

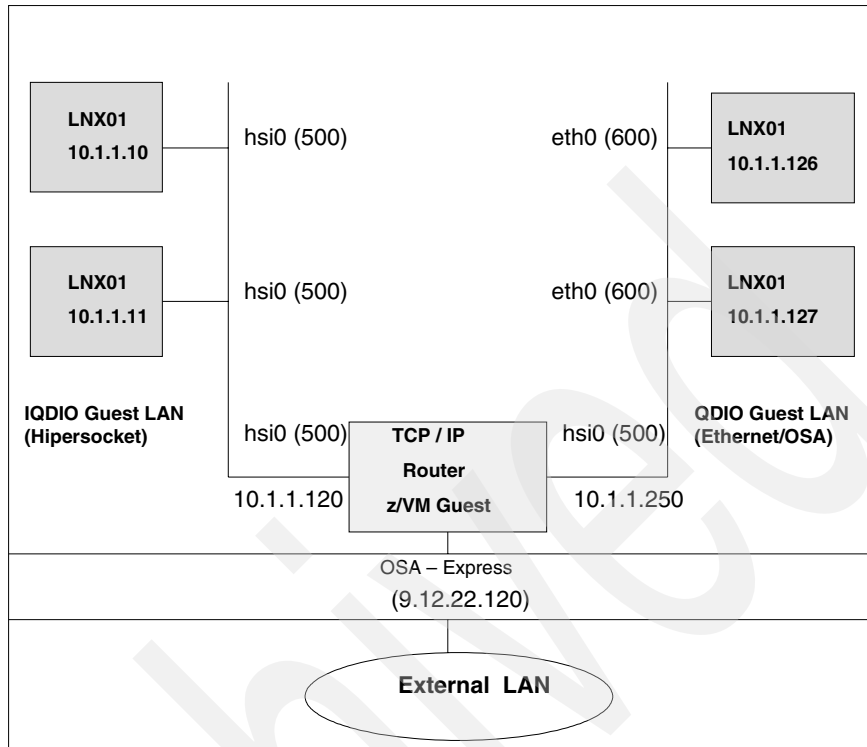


Figure 2-1 Example guest LAN

The amount of VM storage consumed is dependent upon the number of LANs defined, the number of TCP/IP stacks connected to each LAN, and the number of registered IP addresses. The storage is used to build control blocks and IP index tables similar to the IP lookup tables built by the HiperSockets microcode feature. One IP index table is maintained for each guest LAN defined.

Linux guests connected to a z/VM guest LAN must communicate with the physical network through the TCP/IP service machine or a Linux router. This adds both latency and increased CPU utilization to the environment.

To connect to a guest LAN, each guest machine requires a virtual network interface card, or NIC, containing three OSA devices (for read, write, and data channels), which allows the connection of 1,024 guest machines or TCP/IP stacks per LAN. From z/VM 4.3 on, the guest LAN can be defined to use either simulated HiperSockets (iQDIO, or internal queued direct input/output, or QDIO) devices or simulated QDIO devices. The QDIO device that is simulated is the OSA-Express Ethernet adapter.

Tip: Though QDIO and HiperSockets guest LANs are different, when it comes to defining the virtual NIC to the Linux guest, they both use the same Linux QDIO and qeth device drivers.

According to the environment, we have to decide whether to deploy QDIO guest LAN or HiperSockets (iQDIO) guest LAN.

Some of the characteristics of QDIO and iQDIO are:

- ▶ QDIO (OSA-Express simulation)
 - IPv4 and IPv6 support
 - Ethernet transport
 - Asynchronous
 - Can be used as an OSA-Express test network
- ▶ iQDIO (HiperSockets simulation)
 - IPv4 and IPv6 support
 - Supports multicast router connections
 - Deploy maximum transmission units (MTUs) larger than 8 K
 - Synchronous
 - Can be used as a HiperSockets test network
 - Slightly smaller path length in CP than QDIO guest LAN

A benefit of iQDIO guest LAN support for communication among virtual machines is that, because z/VM is simulating all aspects of the HiperSockets function, the locking and unlocking of virtual pages in memory can be more precisely controlled when data is transferred between guest operating systems.

Attention: Although the syntax in the following sections is valid, always refer to the relevant level of z/VM reference manual for a complete description the correct command syntax.

2.2.1 Steps involved in setting up a guest LAN

To set up a guest LAN:

1. Define a guest LAN at system initialization in the SYSTEM CONFIG file or by using CP's DEFINE LAN command. In the latter case, the guest LANs are only valid for the life of a z/VM system. In other words, if that system is shut down and then IPLed, the guest LAN is no longer defined (see Example 2-1).

```
DEFINE LAN lanname [OWNERid ownerid] - [TYPE HiperSockets|QDIO]
[MaxCONN INFinite|nnnn] -[MFS 16K|24K|40K|64K]
[UNRESTRicted|RESTRicted] - [ACCOUnTing ON|OFF] -[GRANT userlist]
```


Example 2-1 Defining a guest LAN

```
DEFINE LAN ITSOLAN OWNERID SYSTEM TYPE QDIO
LAN SYSTEM ITSOLAN is created
Ready; T=0.01/0.01 15:33:26
```

2. Add VMLAN statements to the CP SYSTEM CONFIG file to establish system-wide attributes for all z/VM guest LANs that have been defined to the z/VM operating system. Each VMLAN statement specifies a different system wide attribute and operands.

```
VMLAN LIMit PERSistent INFinite|maxcount
VMLAN LIMit TRANSient INFinite|maxcount
VMLAN ACNT|ACCOUNTing SYSTEM ON|OFF
VMLAN ACNT|ACCOUNTing USER ON|OFF
VMLAN MACprefix nnnnnn
```

3. Create a virtual NIC for each guest machine. After it is defined, this NIC can be connected to the guest LAN. The NIC can be defined permanently through a user directory statement or temporarily (for the life of the guest's session) through a CP command. To create a virtual NIC that will remain permanently defined to a VM guest machine (that is, across guest sessions and across IPLs of the z/VM operating system), use the NICDEF statement in the z/VM user directory. To the guest operating system, the NIC devices look like a range of OSA devices. The NICDEF statement defines virtual devices that are fully simulated by CP. The NIC automatically joins the guest LAN when the z/VM user ID is logged on.

```
NICDEF vdev [TYPE HiperSockets|QDIO] [LAN owner name] [DEvices devs]
[CHPID nn] [MACID nnnnnn]
```

To create a virtual NIC that will only last for the life of a guest, issue the following CP command (see Example 2-2).

```
DEFINE NIC vdev [ operands ]
```

Example 2-2 Defining a network card

```
define nic 600 QDIO
NIC 0600 is created; devices 0600-0602 defined
Ready; T=0.01/0.01 15:40:39
```

Tip: For production environments, we recommend that you make a permanent entry in the CP SYSTEM CONFIG file. This would make the guest LAN persistent. For test environments, the DEFINE LAN command is perfectly valid because it provides the flexibility to dynamically create guest LANs as and when required.

4. Connect the virtual NIC to the guest LAN. As discussed above, if we had used the NICDEF user directory statement to define our NIC, the guest machine automatically connects to the LAN whenever it is logged on.

However, if we chose to dynamically create a NIC card using the DEFINE NIC command, we have an additional step to perform before the device is connected to the guest LAN. The COUPLE CP command is required to attach the virtual NIC to a compatible guest LAN (see Example 2-3).

```
COUPLE vdev T0 [ operands ]
```

Example 2-3 Connecting the NIC with guest LAN

```
couple 0600 to system ITSOLAN
NIC 0600 is connected to LAN SYSTEM ITSOLAN
Ready; T=0.01/0.01 15:42:14
```

Now that we have built a guest LAN, we can use the CP QUERY LAN command to verify the status of the LAN as shown in Example 2-4.

Example 2-4 Verifying the Guest LAN setup

```
CP QUERY LAN
LAN SYSTEM ITSOLAN    Type: QDIO    Connected: 1    Maxconn: INFINITE
PERSISTENT UNRESTRICTED IP Accounting: OFF
Ready; T=0.01/0.01 15:44:11
```

5. After the Linux guest has been booted, configure the appropriate device drivers in that guest to connect to the guest LAN. Because z/VM creates a virtual network adapter (the NIC) that simulates a real OSA-Express or HiperSockets adapter, the configuration on Linux for a virtual adapter is the same as it is for a real one.

Important: A virtual NIC can only be coupled to a compatible guest LAN. For example, a QDIO NIC cannot be coupled to a guest LAN of type HiperSockets.

2.2.2 Benefits of the guest LAN in an integrated environment

Guest LANs are most suited for environments where there is a limitation on real network hardware.

Internet and application service providers can potentially deploy an instance of Linux on System z for each client account and connect the instance to the guest LAN, without the need to buy and configure separate network hardware.

Guests LANs are beneficial:

- ▶ In a production Web serving environment where the heaviest network traffic is anticipated to be between z/VM guests.
- ▶ When running large numbers of guests in a single z/VM image.
- ▶ In development and test environments that can make use of user-owned transient LANs.
- ▶ When isolation of network activity between the virtual and physical network and/or between z/VM guests is required.

2.3 VSWITCH

The z/VM Virtual Switch (VSWITCH) introduced with z/VM V4.4 builds on the guest LAN technology that was delivered in earlier z/VM releases. VSWITCH connects a guest LAN to an external network using an OSA-Express interface. Two additional OSA-Express devices can be specified as backups in the VSWITCH definition. The Linux guests connected to the VSWITCH are on the same subnet as the OSA-Express interface or interfaces and other machines connected to that physical LAN segment.

The z/VM V4.4 implementation of VSWITCH operates at Layer 3 (network layer) of the OSI model. It only supports the transport of IP packets. In other words, it can be used only for TCP/IP applications. All destinations are identified as IP addresses, thus no Media Access Control (MAC) addresses are used (link layer independent), and Address Resolution Protocol (ARP) processing is offloaded to the OSA-Express adapter.

Figure 2-2 shows an example VSWITCH.

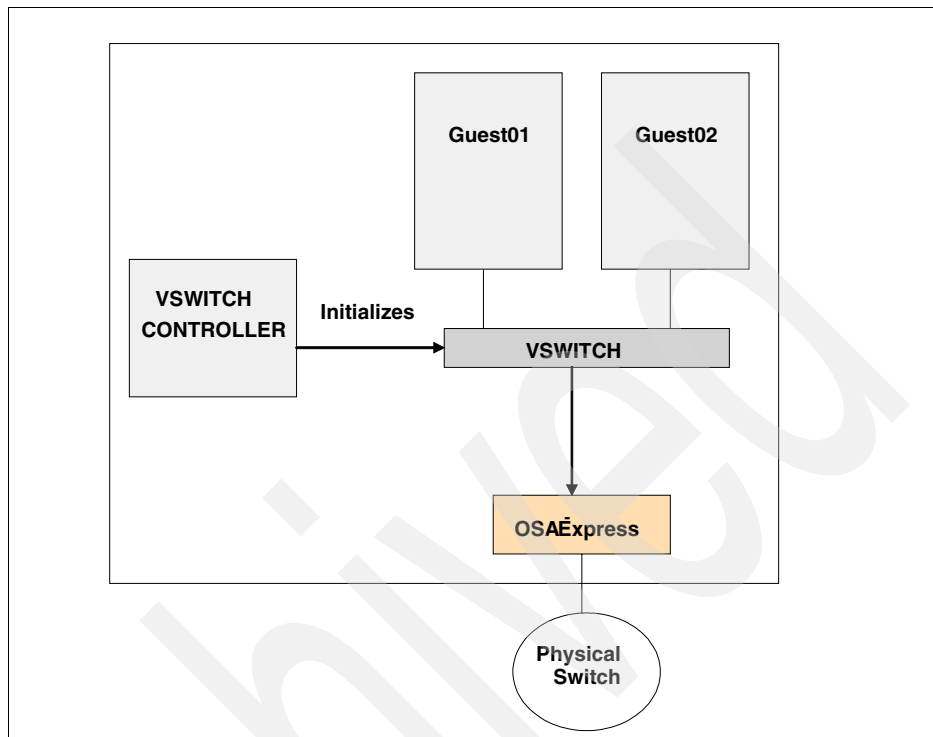


Figure 2-2 Example VSWITCH

In z/VM V5.1, VSWITCH has been enhanced to operate in one of two modes. In IP mode, the VSWITCH operates at Layer 3 (network layer) of the OSI model just as it did in z/VM V4.4

In Ethernet mode, VSWITCH operates at Layer 2 (data link layer) of the OSI model. When operating in Ethernet mode, each guest has a unique MAC address that VSWITCH uses to forward frames. Through the ARP processing of each guest, the guest's MAC address is stored in the ARP cache of hosts residing on the physical side of the LAN segment. Generation and assignment of the locally defined MAC address is performed by z/VM under the control of the LAN administrator.

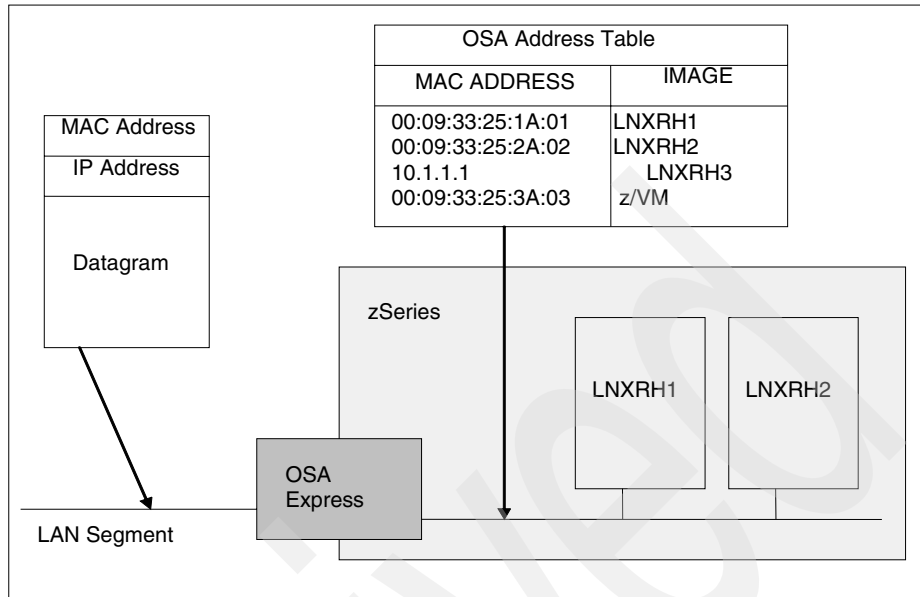


Figure 2-3 Layer 2

Each outbound or inbound frame through the OSA-Express switch trunk connection is an Ethernet frame with the guest's MAC address as the source or destination MAC address. Because the VSWITCH is essentially connected to the physical LAN, the requirement for an intermediate router between the physical and (internal) guest LAN segments is removed. This reduces network latency. In some test cases, it also reduces overall CPU consumption by as much as 30%. Removing the router also means that you no longer require specialized skills to configure and administer a VM-based or Linux-based router.

VSWITCH provides a way to link an external network to guests under z/VM via an OSA Express, without the need for a routing function. Guests attached to the VSWITCH appear to be attached to the LAN that the OSA Express is attached to. This allows you to configure your guests with IP addresses from the network that the OSA Express connects to, without the need to configure proxy ARP in a z/VM TCP/IP service machine.

z/VM Virtual Switch can also function in a disconnected mode, where either an OSA port is not associated, or the associated OSA does not flow traffic to the external network. It might seem that a VSWITCH without an OSA is just the same as a QDIO guest LAN.

2.3.1 VSWITCH configurations

The following steps are required to implement VSWITCH:

- 1. Configure one or more z/VM TCP/IP virtual machines to act as controllers for the VSWITCH.
- 2. Define a VSWITCH to act as a LAN segment for the virtual machines.
- 3. Create a simulated NIC on each virtual machine.
- 4. Link the Linux system to the VSWITCH.

Configuring controller service machines

z/VM Virtual Switch uses a TCP/IP service machine to act as the interface to an OSA Express network connection. This TCP/IP service machine acts as a controller for the VSWITCH and manages the operation of the OSA Express adapter ports the VSWITCH uses. In order for a VSWITCH to provide connectivity to a LAN, at least one TCP/IP service machine must be configured to be a controller.

The configuration allows the TCP/IP stack to connect to the system service that manages VSWITCH connections. The TCP/IP service machine is then able to act as a controller for a VSWITCH OSA connection. Using the CP QUERY CONTROLLER command, you can see the TCP/IP stacks that can serve as VSWITCH controllers. Example 2-5 shows the output from the command.

Example 2-5 Listing VSWITCH controllers

CP QUERY CONTROLLER			
Controller VSWCTL1	Available: YES	VDEV Range: *	Level 520
Capability: IP ETHERNET VLAN_ARP			
SYSTEM VSWITCH1	Primary	Controller: *	VDEV: 2044
Controller VSWCTL2	Available: YES	VDEV Range: *	Level 520
Capability: IP ETHERNET VLAN_ARP			
Ready; T=0.01/0.01 15:47:44			

Any TCP/IP service machine that you want to operate as a controller for a VSWITCH must have the IUCV *VSWITCH statement added to its user directory entry, as shown in Example 2-6. This authorizes the TCP/IP service machine to connect to the *VSWITCH system service.

Example 2-6 VSWITCH user definition

USER VSWCTL1	VSWCTL1	99M	128M	G
INCLUDE IBMDFLT				
IUCV *VSWITCH MSGLIMIT 65535				
OPTION QUICKDSP SVMSTAT MAXCONN 1024 DIAG98 APPLMON				

```
SHARE RELATIVE 3000
LINK TCPMAINT 0591 0591 RR
LINK TCPMAINT 0592 0592 RR
LINK TCPMAINT 0198 0198 RR
MDISK 0191 3390 81 10 LX1USR
```

For a z/VM TCP/IP service machine to initialize as a candidate to control an OSA for a VSWITCH, the VSWITCH CONTROLLER statement must appear in the PROFILE TCPIP for that stack (Example 2-7).

Example 2-7 Defining VSWITCH Controller

```
VSWITCH CONTROLLER ON vdev1 vdev2
VSWITCH CONTROLLER ON
```

Important: We recommend that you define at least two new TCP/IP service machines dedicated to this purpose for backup and isolation reasons. Do *not* add this function to your primary TCP/IP service machine. In Example 2-5 on page 30, we define two virtual machines to act as VSWITCH controllers.

Defining a VSWITCH

A VSWITCH is created using the CP DEFINE VSWITCH command from a z/VM Class B user ID. This creates a VSWITCH called VSWITCH1. It uses OSA device, 2044 to 2046. We also granted Linux guest LNXRH1 access to the VSWITCH. See Example 2-8 for an example of this command.

Example 2-8 Defining VSWITCH

```
DEFINE VSWITCH VSWITCH1 RDEV 2044
MODIFY VSWITCH VSWITCH1 GRANT LNXRH1
```

Before connecting to a Virtual Switch, a guest must first be authorized to the VSWITCH. Authorization checking is done when the guest attempts to connect to the Virtual Switch using the COUPLE command.

Authorization is granted using the SET VSWITCH command with the GRANT option. For example, to authorize guest LNXRH1 access to Virtual Switch VSWITCH1, use the command shown in Example 2-9.

Example 2-9 Providing VSWITCH access to Linux image

```
SET VSWITCH VSWITCH1 GRANT LNXRH1
Command complete
```

Defining a simulated Network Interface Card

Example 2-10 shows a simulated NIC being defined on a guest using the CP DEFINE NIC command.

Example 2-10 Defining a network card

```
CP DEFINE NIC 2400 QDIO
NIC 2400 is created; devices 2400-2402 defined
```

When your simulated QDIO NIC is created, connect your simulated NIC using the CP COUPLE command. In Example 2-11, we use the COUPLE command to connect the simulated NIC at device address 2400 to a VSWITCH called VSWITCH1.

Example 2-11 Connecting the NIC with the VSWITCH

```
CP COUPLE 2400 SYSTEM VSWITCH1
NIC 2400 is connected to VSWITCH SYSTEM VSWITCH1
```

Important: The VSWITCH controller is only involved in device initialization. When initialized, all packets pass directly between connection endpoints without passing through the VSWITCH controller.

2.3.2 Configuring a Layer 2 VSWITCH in z/VM

In Ethernet mode, a VSWITCH operates at Layer 2. Each guest connected to the Virtual Switch is assigned a unique MAC address. Assignment of the guest's MAC address is performed by z/VM under the control of the LAN administrator:

- ▶ The VMLAN statement in the SYSTEM CONFIG file provides the MACPREFIX and MACIDRANGE operands to control generation of the system-wide range of MAC addresses.
- ▶ The NICDEF statement in the user directory entry provides the MACID operand to control the specific MAC address assigned to a guest.

Restriction: Port sharing is only supported between Virtual Switches of the same transport mode. Attempting to communicate between a Layer 2 VSWITCH and a Layer 3 VSWITCH sharing the same OSA-Express adapter results in a network time-out. To resolve this, ensure that the Layer 2 VSWITCH and Layer 3 VSWITCH use separate OSA-Express adapters.

2.3.3 Benefits of VSWITCH on an integrated environment

The benefits of VSWITCH on an integrated environment are:

- ▶ VSWITCH is better suited for test and production environments where there is need for direct network connectivity. Guests attached to the VSWITCH appear to be attached to the LAN that the OSA Express is attached to. This allows you to configure your guests with IP addresses from the network that the OSA Express connects to, without the need to configure proxy ARP in a z/VM TCP/IP service machine.
- ▶ It eliminates the need for a router within the virtual network, which results in reduced latency and CPU utilization.
- ▶ VSWITCH is advantageous for environments where there is a need for low overhead hardware/software backup capabilities for high availability.
- ▶ z/VM V5.2 virtualizes a LAN sniffer to capture network traffic on a z/VM guest LAN or VSWITCH. This capability helps an administrator (or owner of the guest virtual machine) to capture network data to resolve virtual networking problems.
- ▶ With the native Linux tracing capability on a VSWITCH, from a Linux guest, traffic can be traced, recorded, and analyzed by existing tools directly from the guest virtual machine.

2.4 HiperSockets Network Concentrator

HiperSockets Network Concentrator (HSNC) is a combination of tools for enhanced HiperSockets connectivity. HSNC enables integration of nodes that have HiperSockets interfaces to the external network within the same subnet, without any network routing overhead.

In other words, HiperSockets interfaced nodes appear as though they are directly connected to the external network. HSNC completely simplifies the network topology and the server consolidation efforts and delivers increased performance.

Figure 2-4 shows a sample configuration of HSNC.

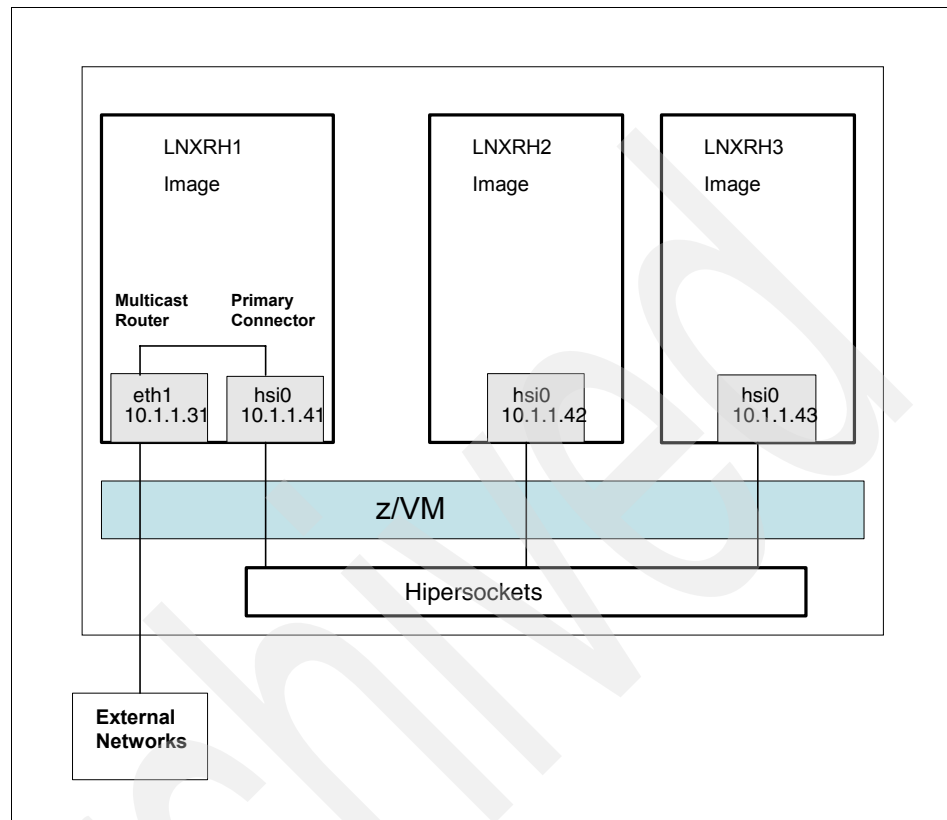


Figure 2-4 Sample configuration of HiperSockets Network Concentrator

A connector Linux system forwards traffic between the external OSA interface and one or more internal HiperSockets interfaces. This is done via IPv4 forwarding for unicast traffic and via a particular bridging code (xcec_bridge) for multicast traffic.

Important: The maximum transmission unit, or MTU, size for the OSA and HiperSockets has to be of same size in the connector system.

A script named `ip_watcher.pl` observes all IP addresses registered in the HiperSockets network and sets them as proxy ARP entries on the OSA interfaces. The script also establishes routes for all internal systems to enable IP forwarding between the interfaces.

2.4.1 Setting up concentrator

In the connector node, the OSA interface for traffic into the LAN must be set as multicast router (see our Example 2-12). This option supports forwarding of all types of packets to the corresponding devices.

Example 2-12 Enabling Multicast router support fto eth0

```
echo multicast_router > /sys/bus/ccwgroup/devices/0.0.f5f0/route4
```

devices	CHPID	interface	cardtype	port	chksum	prio-q'ing	rtr4	rtr6	fsz	cnt
0.0.0600/0.0.0601/0.0.0602	x23	eth0	OSD_1000	0	sw	always_q_2	mc+	no	64k	16

The HiperSockets network has to be configured as a primary connector. For this, the route4 attribute of the corresponding devices has to be set. As in Example 2-13, the HiperSockets interface hsi0 has an IP address of 10.1.1.41

Example 2-13 Enabling Primary router support to hsi0

devices	CHPID	interface	cardtype	port	chksum	prio-q'ing	rtr4	rtr6	fsz	cnt
0.0.e300/0.0.e301/0.0.e302	xE3	hsi0	HiperSockets	0	sw	always_q_2	p+c	no	64k	16
0.0.0600/0.0.0601/0.0.0602	x23	eth0	OSD_1000	0	sw	always_q_2	mc+	no	64k	16
0.0.09a0/0.0.09a1/0.0.09a2	x21	eth1	OSD_100	0	sw	always_q_2	no	no	64k	16

The connector system has to be enabled for IP forwarding. This can be done using the **sysctl** command in Linux, as shown in Example 2-14.

Example 2-14 Enabling IP forwarding

```
sysctl -w net.ipv4.ip_forward=1
```

The network routes for the HiperSockets interface must be removed, and a network route for the HiperSockets Network Concentrator IP subnet has to be established via the OSA interface. To achieve this, the IP address 0.0.0.0 can be assigned to the HiperSockets interface while an address used in the HiperSockets Network Concentrator IP subnet is to be assigned to the OSA interface. This sets the network routes up correctly for HiperSockets Network Concentrator.

2.4.2 Setting up leaf nodes

Leaf nodes do not require any special setups. For the leaf nodes to get connected to the HiperSockets network, they must be configured as though they were directly connected to the LAN. As per the Figure 2-4 on page 34, the leaf nodes are configured with HiperSockets interface hsi0 and are assigned IP addresses 10.1.1.42 and 10.1.1.43 respectively.

2.4.3 Starting the concentrator

To start the HiperSockets Network Concentrator, we have to execute the `start_hsync.sh` script (see Example 2-15). The interface name can be specified as an optional parameter.

Example 2-15 Starting Concentrator script in background processing

```
start_hsync.sh &
```

When the concentrator has been started, verify that the kernel routing table for IP addresses have been added. In Example 2-16, the entries in bold in the routing table are added by the HiperSockets Network Concentrator script.

Example 2-16 Kernel IP table

Kernel IP routing table							
Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
10.1.1.42	*	255.255.255.255	UH	0	0	0	hsi0
10.1.1.41	*	255.255.255.255	UH	0	0	0	hsi0
10.1.1.0	*	255.255.255.0	U	0	0	0	eth0
loopback	*	255.0.0.0	U	0	0	0	lo
default	10.1.1.40	0.0.0.0	UG	0	0	0	eth0

When the IP addresses are added to the kernel routing table by the concentrator, we must be able to ping any image from the external networks.

Important: The MTU of the OSA and HiperSockets link should be of the same size. Otherwise multicast packets not fitting in the link's MTU are discarded as there is no IP fragmentation for multicast bridging.

2.4.4 Stopping the Concentrator

To stop the HiperSockets Network concentrator, we have to issue the `killall ip_watcher.pl` command. This removes all routing table and proxy ARP entries added while using HiperSockets Network Concentrator, as shown in Example 2-17.

Example 2-17 Stopping HiperSockets network concentrator

```
killall ip_watcher.pl
ip_watcher.pl was terminated, cleaning up.
killing xcec-bridge
removing all parp entries from mc interfaces
removing all routes from connector interfaces
```

2.5 High availability

A highly available system is a system that is designed to eliminate or minimize the loss of service due to either planned or unplanned outages.

The zSeries product line is designed to offer layer upon layer of fault tolerance and error checking features. If a failure occurs, the built-in redundancy on the System z platform is intended to shift the work over from failing components to ones that work to prevent the user service from being interrupted. The failed components can be removed and replaced while the processor is still active, so service may continue.

2.5.1 High availability with z/VM and Linux on System z

System z servers running z/VM and Linux on System z can provide important autonomic capabilities and rich function to help provide customers with the tools to make their technology responsive during outages.

Linux on System z can inherit the hardware's reliability and with its strong networking tools base, and we can easily build a resilient and flexible network infrastructure.

Implementing failover pairs to provide network adapter fault tolerance is a simple and effective approach to increase the reliability of server connections. Redundant router configurations, hardware-based and software-based load balancing techniques with switches/routers, and so on, support the implementation of a highly available network infrastructure. Moreover, the network cable can also be a single point of failure.

With Linux on zSeries, there are many tools available to help us to build networks that are highly available. For example:

- ▶ z/VM Virtual Switch can be used to create highly available connectivity for your Linux guests under z/VM. We can make access to the our z/VM guests highly reliable by configuring the redundancy features of VSWITCH combined with LAN-based high availability.
- ▶ We can define multiple OSA Express adapters for hardware redundancy, and multiple TCP/IP controller service machines for some software redundancy.

In this section, we discuss some basic Linux and z/VM tools that are necessary or helpful for implementing a high availability solution for Linux servers running on System z in a z/VM environment.

2.6 Channel bonding

Most Linux distributions allow administrators to bind multiple network interfaces together into a single channel using the bonding kernel module and a special network interface called a *channel bonding interface*. Channel bonding enables two or more network interfaces to act as one, concurrently increasing the bandwidth and providing redundancy.

The Linux bonding driver provides a method for aggregating multiple network interfaces into a single logical *bonded* interface. The behavior of the bonded interfaces depends upon the mode; modes provide either hot standby or load balancing services. Figure 2-5 shows a typical channel bonding configuration.

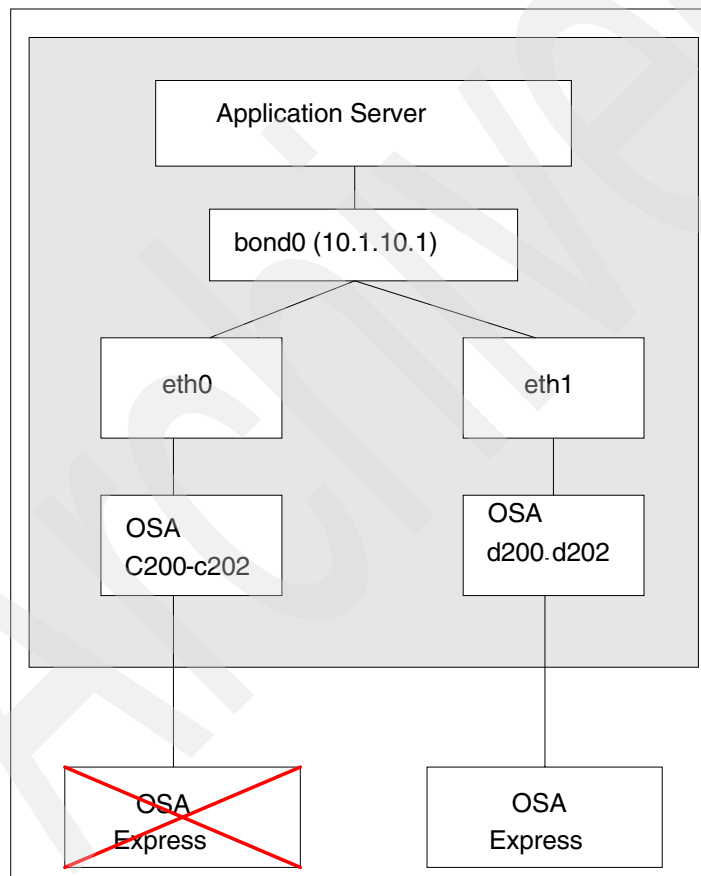


Figure 2-5 Channel bonding configuration

For enabling bonding, the Linux kernel has to be compiled with the “Bonding Driver support” option selected. It is recommended to configure the bonding driver as a module during kernel compilation. Usually modes provide either hot standby or load balancing services. Some of the bonding mode policies:

balance-rr	- Round-robin
active-backup	- Only one slave in the bond will be active
balance-xor	- Transmit based on the selected transmit hash policy
broadcast	- Transmits everything on all slave interfaces
802.3ad	- Dynamic link aggregation
balance-tlb	- Adaptive transmit load balancing
balance-alb tlb + Receive	- Load balancing

Important: If either of the `miimon` or `arp_interval` and `arp_ip_target` parameters are to be specified, there is a possibility of serious network degradation during link failures.

The bonding load balance modes support link monitoring of their members, therefore if individual links fail, the load will be rebalanced across the remaining devices. In Figure 2-5 on page 38, both the network adapters that are connected to `bond0` have to be connected to the same LAN. If the adapters are of different LAN segments, it is better to use normal Layer 3 mode and use dynamic routing software (Quagga).

2.6.1 Configuring bonding devices

Figure 2-3 on page 29 depicts a highly available setup created using bonding in Layer 2 mode. We can configure bonding using either the distributions network initialization scripts, or manually using either `ifenslave` or the `sysfs` interface. For simplicity purposes, we describe the options for configuring bonding using the `ifenslave` interface.

Attention: To configure bonding for different Linux distributions using versions of `initscripts` and `sysconfig`, refer to the bonding documentation available at the bonding project Web site at:

<http://sourceforge.net/projects/bonding>

1. Because our test setup uses direct OSA connectivity, the Ethernet adapters (eth0 /eth1) have to be updated with the MAC address (Example 2-18).

Example 2-18 Specifying MAC address to network devices

```
ifconfig eth0 hw ether 00:09:33:25:1A:01
ifconfig eth1 hw ether 00:09:33:25:2A:02
```

Important: For Direct OSA, the MAC address has to be defined using the command **ifconfig** manually as in Example 2-20. But for VSWITCH, the MAC addresses are created by z/VM VSWITCH.

2. Decide on the options that have to be passed during the bonding module load process. In Example 2-19, we load the bonding module with the miimon options, because we require link failures to be detected.

Example 2-19 Loading bonding kernel module

```
modprobe bonding miimon=100 mode balance-rr
```

3. Using the **ifconfig** command, bring up the bonding devices (Example 2-20).

Example 2-20 Assigning IP address to the bond device

```
ifconfig bond0 10.1.1.1 netmask 255.255.255.0
```

4. Query the bonding configuration (Example 2-21). Each bonding device status can be gathered from /proc/net.bonding/bond0. The bonding configuration, options and state of each slave can be gathered from this file.

Example 2-21 Querying the bonding device

```
Ethernet Channel Bonding Driver: v2.6.3 (June 8, 2005)
Bonding Mode: load balancing (round-robin)
Currently Active Slave: eth0
MII Status: up
MII Polling Interval (ms): 1000
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1
MII Status: up
Link Failure Count: 0
```


Slave Interface: eth0
MII Status: up
Link Failure Count: 0

5. Using the **ifenslave** command, attach or connect the Ethernet devices to bond0 (Example 2-22).

Example 2-22 Connecting the Ethernet devices to bond1

```
ifenslave bond0 eth0  
ifenslave bond0 eth1
```

6. The network configuration can be inspected using the **ifconfig** command. Example 2-23 shows the output of the **ifconfig** command to our configuration. Bonding devices will have the MASTER flag set. Bonding slave devices will have the SLAVE flag set. The **ifconfig** output does not contain information about which slaves are associated with which masters.

Example 2-23 Listing of network configurations

```
bond0    Link encap:Ethernet  HWaddr 00:C0:0A:CF:17:01  
         inet addr:XXX.XXX.XXX.YYY  Bcast:XXX.XXX.XXX.255  
Mask:255.255.252.0  
         UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1  
         RX packets:7224794  errors:0  dropped:0  overruns:0  frame:0  
         TX packets:3286647  errors:1  dropped:0  overruns:1  carrier:0  
         collisions:0 txqueuelen:0  
  
eth0     Link encap:Etheret  HWaddr 00:C0:0A:CF:17:01  
         UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1  
         RX packets:2373024  errors:0  dropped:0  overruns:0  frame:0  
         TX packets:1243137  errors:1  dropped:0  overruns:1  carrier:0  
         collisions:0 txqueuelen:100  
         Interrupt:10 Base address:0x1100  
  
eth1     Link encap:Ethernet  HWaddr 00:C0:0A:CF:17:01  
         UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1  
         RX packets:2641766  errors:0  dropped:0  overruns:0  frame:0  
         TX packets:1243487  errors:0  dropped:0  overruns:0  carrier:0  
         collisions:0 txqueuelen:100  
         Interrupt:9 Base address:0x1808
```

In Example 2-23, the bond0 interface is the master (MASTER), while eth0 and eth1 are slaves (SLAVE). Notice that all slaves of bond0 have the same MAC address (HWaddr) as bond0 for all modes except TLB and ALB that require a unique MAC address for each slave.

Restriction: Currently, due to implementation restrictions in the bonding driver, it is not possible to enable both ARP and MII monitoring simultaneously. This restriction is specific to channel bonding driver version v2.6.3.

2.6.2 Benefits of channel bonding

The benefits of channel bonding are:

- ▶ Easy and flexible to set up failover and/or load-balancing functionality on as network infrastructure.
- ▶ In production environments, bonding can be used as a technique to guard against switch (Layer 2) and cable (Layer 1) failures. Here for example, a machine can be configured with multiple physical connections to separate switch devices while presenting a single logical interface to userspace.
- ▶ It is transparent to LAN infrastructure.
- ▶ Provides better performance depending on the bonding mode.

2.7 VIPA with dynamic routing

Virtual IP address (VIPA), provides an IP address that is owned by a TCP/IP stack, but not associated with any particular physical adapter. Because the VIPA is associated with a virtual device, it is always available as long as the TCP/IP stack is functioning and accessible.

A virtual IP address also eliminates a host's dependency upon individual network interfaces. Incoming packets are sent to the system's VIPA address, but all packets travel through the real network interfaces. This is useful for large systems such as System z machines that have multiple IP adapters, including OSA-Express and HiperSockets. As long as one IP adapter is working and connected to the IP network, others can fail without disrupting the services.

Previously, if an interface failed, any connections to that interface were lost. With VIPA on your system and routing protocols within the network providing automatic reroute, recovery from failures occurs without disruption to the existing user connections that are using the virtual interface as long packets can arrive through another physical interface.

Systems running VIPA are more highly available because adapter outages no longer affect active connections. Because multiple physical adapters carry the system IP traffic, overall load is not concentrated on a single adapter and associated subnet.

In Linux for System z, VIPA function is transparent to the network equipment. No special network equipment or other hardware is needed.

Here are the main steps to set up VIPA on Linux:

1. Create a dummy device using the Linux `ip netns` command.
2. Assign a virtual IP address to the dummy device.
3. Ensure that your service (for example, application server) listens to the virtual IP address assigned above.
4. Set up routes to the virtual IP address, on clients or gateways. We can use either:
 - Static routing
 - Dynamic routing using Quagga; see 2.7.1, “Dynamic routing” on page 44

Setting up VIPA using an example high availability scenario is discussed in the subsequent chapters.

Figure 2-6 shows an example of VIPA with dynamic routing.

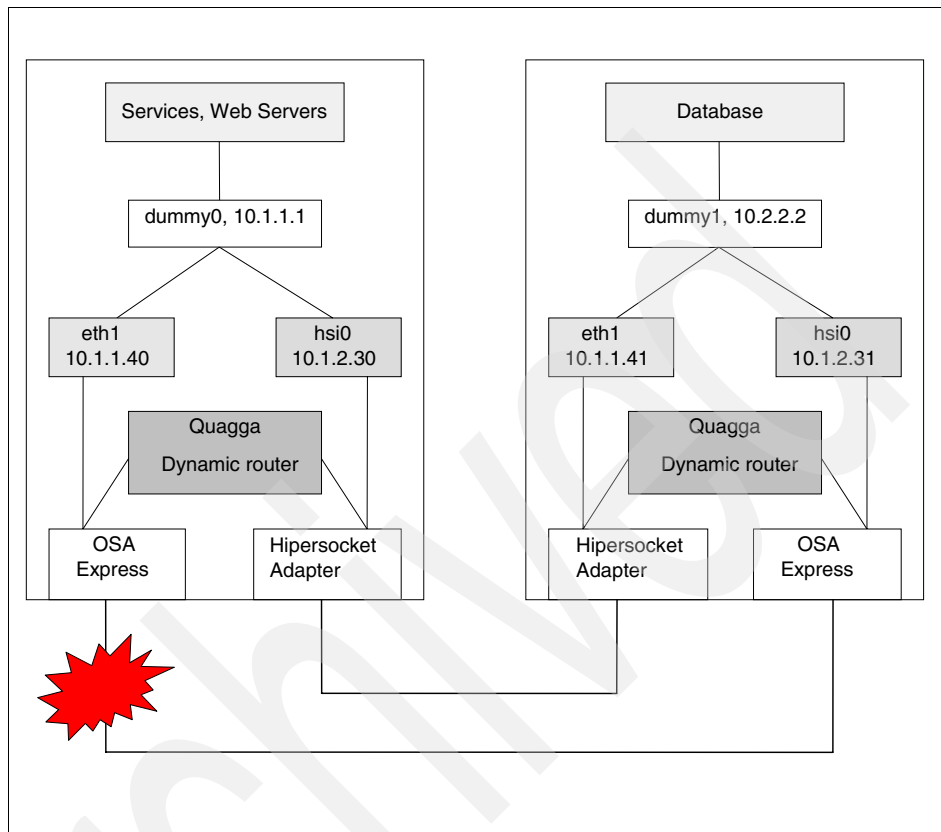


Figure 2-6 VIPA with dynamic routing example

2.7.1 Dynamic routing

Dynamic routing protocols are used primarily to advertise the VIPA address to the networks around. We used a tool called *Quagga*, which assists in enabling dynamic routing.

Quagga is a routing software package that provides TCP/IP based routing services. When Quagga is installed and configured, the system acts a dedicated router. You can find more about Quagga at:

<http://www.quagga.net/about.php>

Using the routing protocols, the machines can exchange routing information with other routers that use the same protocols. Quagga uses this exchanged information to update the kernel routing tables.

The following example assumes dynamic routing is going to be used, and demonstrates:

- ▶ How to configure VIPA
- ▶ How to set up dynamic routing using Quagga

2.7.2 VIPA configuration

To configure VIPA:

1. Create a dummy device (Example 2-24).

Example 2-24 Loading dummy kernel module

```
insmod dummy
```

2. Assign a virtual IP address to the dummy device (Example 2-25).

Example 2-25 Assigning IP address to the dummy device

```
ifconfig dummy0 10.2.2.2
```

3. Qethconf is a specialized script used to configure IBM qeth functions IPA, VIPA and proxy ARP. Using qethconf, enable the network devices for this VIPA so that it accepts packets for this IP address. This is necessary only on OSA-Express devices in QDIO mode (Example 2-26).

Example 2-26 Configuring VIPA

```
qethconf vipa add 10.1.1.1 eth1  
qethconf vipa add 10.1.1.1 hsi0
```

4. Verify that the virtual IP address has been correctly set up (Example 2-27).

Example 2-27 Listing of configured VIPA

```
qethconf vipa list  
vipa add 10.1.1.1 eth1  
vipa add 10.1.2.30 hsi0
```

Repeat the above steps on your second system (in our case, LNXRH2) with appropriate IP addresses and VIPA configurations as in Figure 2-4 on page 34.

2.7.3 Dynamic routing configuration

Here we configure the zebra (Example 2-28) and ospfd daemons (Example 2-29) to activate the dynamic routing facility for our scenario. The zebra daemon is an IP routing manager. It provides kernel routing table updates, interface lookups, and redistribution of routes between different routing protocols. Each daemon has its own config files, which has to be properly updated with configuration details.

Example 2-28 Sample configuration details of /etc/quagga/zebra.conf

```
hostname quagga
password quagga
enable password quagga
log file /var/log/quagga/quagga.log
```

The other daemon that we are going to use from the package is OSPF V2. Open Shortest Path First (OSPF) is a routing protocol, which provides scalable network support and faster convergence time. OSPF is widely used in large networks such as ISP backbones and enterprise networks.

Example 2-29 Sample configuration details of /etc/quagga/ospfd.conf

```
hostname quagga
password quagga
enable password quagga
router ospf
area 1.1.1.1 stub
network 10.1.1.0/24 area 1
network 10.1.2.0/24 area 1
network 10.2.0.0/24 area 1
interface eth1
interface hsi0
interface dummy0
```

2.7.4 Starting dynamic routing daemons

When the configuration details are furnished, the daemons can be started. We invoke zebra with the “-dk” option, which specifies that zebra will run in daemon mode (other options are batch mode) and will not delete old self-inserted routes (Example 2-30).

Example 2-30 Starting zebra and ospf daemons

```
./zebra -dk
./ospfd -d
```

Repeat the zebra and ospfd setups on LNXRH2 system with appropriate IP addresses and VIPA configurations as seen in Figure 2-4 on page 34.

Important: Because ospfd gathers interface information from zebra, care needs to be taken to start zebra before invoking ospfd.

2.7.5 Verifying the router

When the daemons are started on both the Linux images, verify the routing table. The routers will have updated the table with *dummy0* as the gateway for some IP addresses (Example 2-31).

Example 2-31 Routing table of LNXRH1 after starting the dynamic routing daemons

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
9.152.24.0	*	255.255.252.0	U	0	0	0	eth0
10.1.1.0	*	255.255.255.0	U	0	0	0	eth1
10.1.2.0	*	255.255.255.0	U	0	0	0	hsi0
10.2.0.0	10.1.1.41	255.255.0.0	UG	20	0	0	eth1 <-- Added by Quagga
10.3.0.0	*	255.255.0.0	U	0	0	0	dummy0
link-local	*	255.255.0.0	U	0	0	0	eth0
loopback	*	255.0.0.0	U	0	0	0	lo
default		192.168.70.1	0.0.0.0	UG	0	0	0 eth0

Now we are able to communicate between both the linux images (LNXRH1 and LNXRH2) using the dummy0 IP address respectively. We have to ensure that our services (for example, applications and databases) listen to the virtual IP address assigned. Randomly bring down one of the network adapters in LNXRH1 using the following command:

```
ifdown eth1
```

In our test setup, even when one of the network adapters was down, the communication between the two Linux images were not affected. This is because Quagga will have renegotiated the route and the network transfer will be taking place on the other adapter; in our case, it is the HiperSockets adapter (hsi0). See Example 2-32.

Example 2-32 Routing table of LNXRH1 after eth1 is brought down

Kernel IP routing table									
Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface		
192.168.70.0	*	255.255.255.0	U	0	0	0	eth0		
10.1.1.0	10.1.2.31	255.255.255.0	UG	20	0	0	hsi0		
10.1.2.0	*	255.255.255.0	U	0	0	0	hsi0		
10.2.0.0	10.1.2.31	255.255.0.0	UG	20	0	0	hsi0	<-	New Route added
10.2.0.0	10.1.1.41	255.255.0.0	UG	20	0	0	eth1		
10.3.0.0	*	255.255.0.0	U	0	0	0	dummy0		
link-local	*	255.255.0.0	U	0	0	0	eth0		
loopback	*	255.0.0.0	U	0	0	0	lo		
default	192.168.70.1	0.0.0.0	UG	0	0	0	eth0		

From Example 2-32, it is evident that during a network adapter failure, VIPA combined with dynamic routing is able to alter between the network connectivities.

Attention: Although we are restricted to our test lab environment, effort is taken to reflect the reality. The scope of this scenario is restricted only to the brief concepts, because the detailed technical information is already described under different topics in various other IBM Redbooks.

Resource management under z/VM

This chapter describes how to manage the system resources such as CPU, memory, and input/output (I/O) in a guest environment running on z/VM. This chapter also explains the power of z/VM to emulate hardware resources for purposes such as testing, and so on, which does not exist on the real physical system.

In this chapter, we briefly describe how to dynamically change the system resources for a z/VM virtual machine. We also demonstrate how to recognize these changes from a Linux system running as the guest operating system.

In this chapter, we discuss the following topics:

- ▶ Importance of resource management under z/VM
- ▶ Resources that can be changed dynamically under z/VM
- ▶ Managing dynamic direct access storage device (DASD) changes under z/VM
- ▶ Managing dynamic network changes under z/VM
- ▶ Managing dynamic CPU changes under z/VM
- ▶ Storage management under z/VM
- ▶ Resource simulation using z/VM

3.1 Importance of resource management under z/VM

The resource management under z/VM is very important because z/VM uses the real resources assigned to it to create and control *virtual machines*, control real devices, and simulate devices for use of virtual machines. The Control Program (CP) is the component that does the resource balancing act on behalf of the z/VM.

The virtual machines created on z/VM are the functional equivalent of real computer systems. The resources seen by the virtual machines running on z/VM are virtual resources. The main resources that are virtualized for virtual machines by z/VM are processor, storage, console, and I/O.

It is also important to note that z/VM is capable of simulating devices or an environment that does not exist physically. One example is simulating a virtual Parallel Sysplex without a real coupling facility (CF) and real coupling links (CL).

3.2 Resources that can be changed dynamically under z/VM

Dynamic management of resources for a virtual machine depends on how CP virtualizes these resources. CP does support Channel Report Word (CRW) virtualization that tells the guest a device has been attached, detached, or changed. This means it is possible to make dynamic I/O changes on a virtual machine.

CP does not virtualize the architectural change notifications that are required to allow dynamic memory and CPU changes. However, we describe some best practices to manage CPU and memory requirements.

Dynamic resource management also depends on the guest operating system. If the guest operating system is not capable of managing a dynamic resource change, then the resource change does not take effect until an IPL of the guest operating system even if z/VM supports the dynamic change. Table 3-1 compares the dynamic resource support under different operating systems that are supported as a guest operating system under z/VM.

Table 3-1 Comparison of dynamic resource management capabilities

Resource	Guest operating system				
	Linux	z/OS	z/TPF	z/VM	z/VSE
CPU	Yes	Yes	No	Yes	Yes
Storage	No	No	No	No	No
DASD	Yes	Yes	Yes	Yes	Yes ^a
Network	Yes	Yes	Yes	Yes	Yes ^a

a. If the device address is predefined at IPL time, then the device can be added dynamically.

We examine managing each of these resources from z/VM in the sections to follow. Linux is used as an example guest operating system to demonstrate dynamic resource management from a guest operating system point of view.

3.3 Managing dynamic DASD changes under z/VM

This section discusses the dynamic addition and removal of DASD devices for both z/VM and Linux environments.

3.3.1 Dynamically adding a DASD device to a z/VM virtual machine

The following steps outline the process of dynamically adding a DASD device to a z/VM virtual machine.

1. Understand the current configuration of the z/VM virtual machine.

We explain the dynamic DASD management under z/VM with an example. Consider the directory entry shown in Example 3-1.

Example 3-1 Sample z/VM user directory entry

```

USER LNX001 LNX001 512M
INCLUDE LINDFLT
MDISK 0191 3390 0001 0080 CMS001 MR
MDISK 0200 3390 0001 3338 LIN001 MR
MDISK 0201 3390 0001 3338 LIN002 MR

```

If we log in to the virtual machine “LNX001” described in Example 3-1, we can see three DASD devices as expected. Example 3-2 shows the QUERY DASD output from this virtual machine.

Example 3-2 QUERY DASD command output

```
DASD 0190 3390 520RES R/O 107 CYL ON DASD E600 SUBCHANNEL = 0031
DASD 0191 3390 CMS001 R/W 80 CYL ON DASD E621 SUBCHANNEL = 0001
DASD 019D 3390 520W01 R/O 146 CYL ON DASD E601 SUBCHANNEL = 0032
DASD 019E 3390 520W01 R/O 250 CYL ON DASD E601 SUBCHANNEL = 0033
DASD 0200 3390 LIN001 R/W 1000 CYL ON DASD E610 SUBCHANNEL = 0002
DASD 0201 3390 LIN002 R/W 3300 CYL ON DASD E347 SUBCHANNEL = 0003
```

Note: DASD devices 019D and 019E are defined in the profile “LINDFLT” and are outside the scope of this discussion.

2. Define a new DASD device and place the z/VM directory online.

While the virtual machine continues to be in the operating state (that is, logged in), we add a new DASD device to its directory entry and place the new directory online. The new directory entry is shown in Example 3-3.

Example 3-3 The new directory entry for the LNX001 z/VM user

```
USER LNX001 LNX001 512M
INCLUDE LINDFLT
MDISK 0191 3390 0001 0080 CMS001 MR
MDISK 0200 3390 0001 3338 LIN001 MR
MDISK 0201 3390 0001 3338 LIN002 MR
MDISK 0202 3390 0001 3338 LIN003 MR
```

The entry for the new DASD device (0202) is highlighted in blue color in Example 3-3.

3. Use the LINK command to effect the change in the z/VM virtual machine.

The virtual machine still sees only the three DASD devices, which it found when it was started (that is, when the virtual machine was logged in). To make the virtual machine aware of the new DASD device dynamically (that is, without logging off and logging back into z/VM), we can use the LINK command. When the LINK command is successfully completed, the new DASD device is visible from the virtual machine. Example 3-4 shows the LINK command and the QUERY DASD command outputs.

Example 3-4 LINK and corresponding QUERY DASD command outputs

```
link * 202 202
READY; T=0.01/0.01 22:50:08
```

query dasd

```
DASD 0190 3390 520RES R/O 107 CYL ON DASD E600 SUBCHANNEL = 0031
DASD 0191 3390 CMS001 R/W 80 CYL ON DASD E621 SUBCHANNEL = 0001
```

```
DASD 019D 3390 520W01 R/O 146 CYL ON DASD E601 SUBCHANNEL = 0032
DASD 019E 3390 520W01 R/O 250 CYL ON DASD E601 SUBCHANNEL = 0033
DASD 0200 3390 LIN001 R/W 1000 CYL ON DASD E610 SUBCHANNEL = 0002
DASD 0201 3390 LIN002 R/W 3300 CYL ON DASD E347 SUBCHANNEL = 0003
DASD 0202 3390 LIN003 R/W 3300 CYL ON DASD E348 SUBCHANNEL = 0003
```

The entry for the new DASD device (0202) is highlighted in blue color.

3.3.2 Dynamically removing a DASD device from a z/VM virtual machine

The following steps outline the process of dynamically removing a DASD device from a z/VM virtual machine.

1. Remove the DASD from the z/VM guest operating system.

If there is a guest operating system running in the virtual machine, the DASD device must be removed, gracefully, from that guest operating system before removing from the virtual machine level.

For an example, refer to 3.3.4, “Dynamically removing a DASD device from a z/VM Linux guest” on page 54, for dynamically removing a DASD device from a Linux guest running on z/VM.

2. Remove the DASD device from the z/VM virtual machine.

When the DASD device has been removed from the guest operating system running in the virtual machine, the DETACH command can be used to remove the DASD from the virtual machine. Example 3-5 shows the DETACH command output and the corresponding QUERY DASD output.

Example 3-5 DETACH and corresponding QUERY DASD command outputs

detach 202

DASD 0202 DETACHED

READY; T=0.01/0.01 23:14:04

query dasd

```
DASD 0190 3390 520RES R/O 107 CYL ON DASD E600 SUBCHANNEL = 0031
DASD 0191 3390 CMS001 R/W 80 CYL ON DASD E621 SUBCHANNEL = 0001
DASD 019D 3390 520W01 R/O 146 CYL ON DASD E601 SUBCHANNEL = 0032
DASD 019E 3390 520W01 R/O 250 CYL ON DASD E601 SUBCHANNEL = 0033
DASD 0200 3390 LIN001 R/W 1000 CYL ON DASD E610 SUBCHANNEL = 0002
DASD 0201 3390 LIN002 R/W 3300 CYL ON DASD E347 SUBCHANNEL = 0003
```

The output confirms that the DASD device 0202 is removed from the virtual machine.

3.3.3 Dynamically adding a DASD device to a z/VM Linux guest

The following steps outline the process of dynamically adding a DASD device to a z/VM Linux guest.

1. Add the DASD device to the z/VM virtual machine.

The DASD device must be added to the z/VM virtual machine running the Linux guest before Linux can recognize the same. Refer to 3.3.1, “Dynamically adding a DASD device to a z/VM virtual machine” on page 51.

2. Bring the DASD device online from the z/VM Linux guest.

When the a DASD device is dynamically added to the virtual machine, it can be added to the z/VM Linux guest as shown in Example 3-6.

Example 3-6 Dynamically bringing a DASD device online from Linux

for linux 2.4 kernel based systems:

```
[root@lhx001 ~]# echo "add 202" > /proc/dasd/devices
```

for linux 2.6 kernel based systems:

```
[root@lhx001 ~]# echo "1" > /sys/bus/ccw/devices/0.0.0202/online
```

The command result can be verified by examining either the **dmesg** command output or by examining the **tail /var/log/messages** command output from the Linux guest. Alternatively, running the **cat /proc/dasd/devices** command displays the entire list of DASD devices recognized by the Linux guest.

3.3.4 Dynamically removing a DASD device from a z/VM Linux guest

The following steps outline the process of dynamically removing a DASD device from a z/VM Linux guest.

1. Unmount the file systems from Linux.

Before bringing the DASD device offline from the Linux guest, any I/O activity on that DASD device must be stopped. Bringing the DASD device offline without stopping the I/O activity on it can be disastrous. If there are file systems created on the DASD device and are mounted, then *all* those file systems must be unmounted. Unmounting a file system might not be easy depending on where it is mounted and what processes are accessing it. If the file system is a system partition such as **/**, **/usr**, **/tmp**, **/var**, and so on, then the unmounting might not be possible while the guest operating system is running. Also, these partitions host the operating system installation and removing them means removing the operating system itself.

If the file system is a data partition, then it might be possible to unmount the partition if there is no process accessing that partition and is not needed by any applications currently running on the system.

Therefore, in general, unmounting the file systems has to be done with care and it may or may not be possible depending on the type of partition. The command to unmount a file system from Linux is **umount**. The situation gets more complicated when the DASD device is controlled by a Logical Volume Manager (LVM) or configured as part of a Linux Redundant Array of Independent Disks (RAID) array. How to deal with a LVM or a RAID device is beyond the scope of this book.

2. Bring the DASD device offline from a z/VM Linux guest.

When the I/O activity on a DASD device is gracefully stopped, it can be made offline from Linux by using the procedure shown in Example 3-7.

Example 3-7 Dynamically bringing a DASD device offline from Linux

for linux 2.4 kernel based systems:

```
[root@lnx001 ~]# echo "set 202 off" > /proc/dasd/devices
```

for linux 2.6 kernel based systems:

```
[root@lnx001 ~]# echo "0" > sys/bus/ccw/devices/0.0.0202/online
```

3.4 Managing dynamic network changes under z/VM

The main network resource that has to be defined for a z/VM virtual machine for it to communicate in the network is the virtual network interface card (NIC). Most of the other networking-related resources are handled by z/VM directly and are generally transparent to the virtual machines running on z/VM. For example, consider the virtual switch failover. The virtual machine or the guest operating system is not affected by a virtual switch failover and does not see a drop in the network connection.

In this section, we demonstrate how to dynamically add and remove a NIC to and from a z/VM virtual machine.

Note: There are several networking scenarios possible on a z/VM system. We assume the usage of the virtual switch for the purpose of our discussion in this chapter.

3.4.1 Dynamically adding a NIC to a z/VM virtual machine

Here are the steps involved in dynamically adding a NIC to a z/VM virtual machine:

1. Understand the current configuration.

The QUERY NIC command can be used to understand the current NICs configured on a z/VM virtual machine. Example 3-8 shows the QUERY NIC command output.

Example 3-8 The QUERY NIC command output

```
query nic
ADAPTER 0006  TYPE: QDIO      NAME: UNASSIGNED  DEVICES: 3
PORT 0 MAC: 02-00-00-00-00-20  VSWITCH: SYSTEM VSWITCH1
```

2. Add the new NIC dynamically to the z/VM virtual machine.

It is very simple to add another NIC to a z/VM virtual machine on the fly. Example 3-9 demonstrates this using the DEFINE and COUPLE commands.

Example 3-9 Adding a NIC dynamically to a z/VM guest

```
define nic 0010 type qdio
NIC 0010 IS CREATED; DEVICES 0010-0012 DEFINED

couple 010 system vswitch1
NIC 0010 IS CONNECTED TO VSWITCH SYSTEM VSWITCH1
```

The DEFINE command creates a NIC of type QDIO and the COUPLE command connects the NIC to the virtual switch “VSWITCH1”.

Note: The number 0010 is an arbitrary number that is not already in use in the same virtual machine.

The QUERY NIC command can be used to verify that the new NIC is added and connected to the VSWITCH1. The QUERY NIC output after coupling the NIC is shown in Example 3-10. The new NIC detail is shown in blue color.

Example 3-10 The QUERY NIC command output after adding the NIC

```
query nic
ADAPTER 0006  TYPE: QDIO      NAME: UNASSIGNED  DEVICES: 3
PORT 0 MAC: 02-00-00-00-00-20  VSWITCH: SYSTEM VSWITCH1
ADAPTER 0010  TYPE: QDIO      NAME: UNASSIGNED  DEVICES: 3
PORT 0 MAC: 02-00-00-00-00-21  VSWITCH: SYSTEM VSWITCH1
```

3. Making the change persistent.

The NIC, which is added in step 2 on page 56, does not appear when the virtual machine is logged off and logged in again. The NIC has to be created again the next time when the virtual machine is logged in to z/VM. To make the change persistent, an appropriate statement for the NIC should be added to the z/VM directory entry for the virtual machine and the new z/VM directory should be placed online. Example 3-11 shows the z/VM directory statement that has to be added.

Example 3-11 z/VM directory statement for adding the NIC

```
NICDEF 0010 TYPE QDIO LAN SYSTEM VSWITCH1
```

3.4.2 Dynamically removing a NIC from a z/VM virtual machine

Here are the steps involved in dynamically removing a NIC from a z/VM virtual machine:

1. Dynamically remove the NIC from the guest operating system.

The NIC must be removed gracefully from the guest operating system before removing from the virtual machine. As an example, refer to 3.4.4, “Dynamically removing a NIC from a z/VM Linux guest” on page 60.

2. Remove the NIC from the z/VM virtual machine.

When the NIC is gracefully removed from the guest operating system, it can be removed from the virtual machine by using the DETACH command as shown in Example 3-12.

Example 3-12 The DETACH command output

```
detach nic 010  
NIC 0010 IS DESTROYED; DEVICES 0010-0012 DETACHED
```

3. Making the change persistent.

The NIC, which is removed in step 2, is added to the virtual machine when it is logged off and logged in to z/VM, if there is a z/VM directory entry for the NIC removed. To make the change persistent, the z/VM directory statement for the removed NIC should be eliminated from the directory entry for the virtual machine. The new z/VM directory should be placed online after making the changes. In this case, the z/VM directory statement to be removed is shown in Example 3-13.

Example 3-13 z/VM directory entry for the removed NIC

```
NICDEF 0010 TYPE QDIO LAN SYSTEM VSWITCH1
```

3.4.3 Dynamically adding a NIC to a z/VM Linux guest

Here are the steps involved in dynamically adding a NIC to a z/VM Linux guest:

1. Add the NIC to the virtual machine.

Before adding the NIC to the Linux operating system, it must be added to the virtual machine running the Linux guest operating system. Refer to 3.4.1, “Dynamically adding a NIC to a z/VM virtual machine” on page 56 for more details.

2. Find the NIC channel device addresses.

Use the QUERY NIC command to determine the channel devices’ addresses. Example 3-14 shows the QUERY NIC output with the channel device address highlighted in blue color for the newly added NIC.

Example 3-14 QUERY NIC output showing the NIC channel device address

```
query nic
ADAPTER 0006 TYPE: QDIO      NAME: UNASSIGNED DEVICES: 3
PORT 0 MAC: 02-00-00-00-00-20 VSWITCH: SYSTEM VSWITCH1
ADAPTER 0010 TYPE: QDIO      NAME: UNASSIGNED DEVICES: 3
PORT 0 MAC: 02-00-00-00-00-21 VSWITCH: SYSTEM VSWITCH1
```

The starting channel device in this case is “0010”. Therefore, the next two channel devices are “0011” and “0012” (consecutive addresses).

3. Add the NIC to the z/VM Linux guest (2.4 kernel).

The NIC can be dynamically added to a Linux 2.4 kernel based z/VM guest system by echoing the channel device address to the /proc/chandev file and running the **reprobe** command thereafter. Example 3-15 shows these two commands.

Example 3-15 Echoing channel device address and reprobing to dynamically add a NIC

```
[root@lnx001 ~]# echo "qeth1,0x0010,0x0011,0x0012" > /proc/chandev
[root@lnx001 ~]# echo reprobe >/proc/chandev
```

Note: The qeth1 represents the “eth1” interface. Based on the Linux guest configuration, this should be substituted with the next available number for the NIC, before running the **echo** command.

The **dmesg** command can be used after the execution of each of the above commands to find the resulting messages. To make the change persistent across reboots, an entry should be added to the /etc/chandev.conf file. The entry that has to be added is shown in Example 3-16.

Example 3-16 Entry to be added to the /etc/chandev.conf file

```
qeth1,0x0010,0x0011,0x0012
```

The new NIC status can be verified by looking at the /proc/chandev file. Example 3-17 shows the /proc/chandev file output. The detail about the new NIC is highlighted in blue color.

Example 3-17 Portion of /proc/chandev file showing the NIC status

```
0x000f 0x0010 0x0011 0x0006 0x0007 0x0008 0x10 0 0x0306de00 eth0
8192    good          good          good
0x001a 0x001b 0x001c 0x0010 0x0011 0x0012 0x10 0 0x3c956c00 eth1
8192    good          good          good
```

4. Dynamically add the NIC to the z/VM Linux guest (2.6 kernel).

To add the NIC to a Linux 2.6 kernel based z/VM guest system, first the qeth driver module has to be loaded using the **modprobe** command (if not loaded already), then the new qeth device is created by grouping the CCW devices and finally, the new device should be brought online by using the **echo** command. Example 3-18 demonstrates each of these tasks.

Example 3-18 Bringing the qeth device online on Linux 2.6 kernel based z/VM guest

```
[root@lnx001 ~]# modprobe qeth
[root@lnx001 ~]# echo "0.0.0010,0.0.0011,0.0.0012" >
/sys/bus/ccwgroup/drivers/qeth/group
[root@lnx001 ~]# echo "1" >/sys/devices/qeth/0.0.0010/online
```

The **dmesg** command output after each command gives the status messages. The new NIC status can be verified by examining the /proc/qeth file contents as shown in Example 3-19. The new NIC detail is highlighted in bold.

Example 3-19 The /proc/qeth file contents showing the NIC detail

```
[root@lnx001 ~]# cat /proc/qeth
```

devices	CHPID	interface	cardtype	port	chksum
prio-q'ing rtr4 rtr6 fsz cnt					
0.0.0006/0.0.0007/0.0.0008	x1C	eth0	GuestLAN QDIO	0	sw
always_q_2 no no 64k 16					
0.0.0010/0.0.0011/0.0.0012	x1D	eth1	GuestLAN QDIO	0	sw
always_q_2 no no 64k 16					

The next time when an IPL of the Linux system is done, the new NIC will be detected automatically. Hence, there is no need to make entries in a file to make the change persistent.

3.4.4 Dynamically removing a NIC from a z/VM Linux guest

Here are the steps involved in dynamically removing a NIC from a z/VM Linux guest:

1. Stop the network interface from Linux guest.

Before removing the NIC, the network interface configured at the Linux level should be stopped. If there is any active network connection to the IP address configured on the interface, then those connections are terminated when the interface is stopped. Example 3-20 shows usage of the **ifconfig** command to stop the eth1 network interface.

Example 3-20 Stopping the network interface from Linux guest

```
[root@lnx001 ~]# ifconfig eth1 down
```

2. Remove the NIC from the z/VM Linux guest.

The method to dynamically remove a NIC from a z/VM Linux guest is shown in Example 3-21.

Example 3-21 Dynamically removing the NIC from a z/VM Linux guest

for Linux 2.4 kernel based systems:

```
[root@lnx001 ~]# echo "shutdown,0x0010" >/proc/chandev
```

for Linux 2.6 kernel based systems:

```
[root@lnx001 ~]# echo "0" >/sys/devices/qeth/0.0.0010/online
```

3. Remove the NIC from the z/VM virtual machine.

Refer to 3.4.2, “Dynamically removing a NIC from a z/VM virtual machine” on page 57, to dynamically remove a NIC from a z/VM virtual machine.

3.5 Managing dynamic CPU changes under z/VM

CP does not virtualize the architectural change notifications that are required to allow dynamic CPU changes. However, it is possible to create a z/VM virtual machine with an upper limit for the number of CPUs specified. The upper limit when defined allows the z/VM virtual machine to define its own CPU as long as the total number of CPUs for that virtual machine does not exceed the upper limit specified in the z/VM directory entry. Example 3-22 shows a sample z/VM directory entry for a virtual machine.

Example 3-22 Sample z/VM directory entry for a virtual machine

```
USER LNX001 LNX001 512M
INCLUDE LINDFLT
MACHINE ESA 2
MDISK 0191 3390 0001 0080 CMS001 MR
MDISK 0200 3390 0001 3338 LIN001 MR
MDISK 0201 3390 0001 3338 LIN002 MR
```

The numeric entry 2 in the MACHINE ESA 2 means that this virtual machine can define a maximum of two virtual processors by its own. When the virtual machine is started (that is, logged on to z/VM), only one CPU will be allocated by default. The virtual machine can then dynamically create one more by using the DEFINE CPU command. Alternatively, place the appropriate entries in the PROFILE EXEC to automate the same. However, it is not possible to alter the upper limit for CPUs for a virtual machine dynamically. Altering the upper limit requires the virtual machine to log off and log in back to z/VM to effect the change.

3.5.1 Dynamically adding a CPU to a z/VM virtual machine

Here are the steps involved in dynamically adding a CPU to a z/VM virtual machine:

1. Check the currently defined CPUs.

To get a list of currently defined CPUs in a z/VM virtual machine, use the QUERY CPUS command. Example 3-23 shows the QUERY CPUS command output. We can see that only one CPU is created for this z/VM virtual machine.

Example 3-23 Using the QUERY CPUS command to list the defined CPUs

```
query cpus
CPU 00 ID FF0B9D1A20848000 (BASE)
READY; T=0.01/0.01 21:18:32
```

2. Add the new CPU from the z/VM virtual machine.

From the z/VM virtual machine, run the DEFINE CPU command as shown in Example 3-24 to add a new CPU to the virtual machine.

Example 3-24 Adding a new CPU using the DEFINE CPU command

```
define cpu 001
00: CPU 01 DEFINED
READY; T=0.01/0.01 21:21:57
```

3. Check the defined CPUs on the z/VM virtual machine again to verify.

Use the QUERY CPUS command again to see a list of defined CPUs in the z/VM virtual machine. Example 3-25 shows the QUERY CPUS output.

Example 3-25 QUERY CPUS output after defining the CPU

```
00: CPU 00 ID FF0B9D1A20848000 (BASE)
00: CPU 01 ID FF0B9D1A20848000 STOPPED
```

We now have two CPUs defined for this z/VM virtual machine. The STOPPED status for the second CPU means that the guest operating system running in this z/VM virtual machine has not yet started using this CPU. Depending on the type of the guest operating system, an IPL may or may not be needed for the guest operating system to start using this CPU. For an example, refer to 3.5.3, “Dynamically adding a CPU to a Linux guest running on z/VM” on page 62 to find how to do this from a Linux guest without an IPL.

3.5.2 Dynamically removing a CPU from a z/VM virtual machine

It is not possible to dynamically remove a CPU from a z/VM virtual machine because the virtual machine storage is cleared and the virtual machine is reset when a CPU is removed.

3.5.3 Dynamically adding a CPU to a Linux guest running on z/VM

Dynamically adding a CPU to a z/VM virtual machine does not necessarily mean that the guest operating system running in that virtual machine is aware of the change. Some guest operating systems require an IPL to recognize the newly added CPU. We demonstrate, as an example, how to dynamically add a CPU to a Linux guest operating system running in a z/VM virtual machine.

1. Add the CPU dynamically to the z/VM virtual machine running Linux.

Before we add a new CPU to a guest operating system, we define it at the virtual machine level. Refer to 3.5.1, “Dynamically adding a CPU to a z/VM virtual machine” on page 61 for more details.

Note: Linux 2.4 kernel does not support dynamic CPU activation.

2. Add the CPU dynamically to the z/VM Linux guest.

When the new CPU is visible from the z/VM virtual machine, it can be dynamically added to a Linux guest running at kernel level 2.6 or later. Example 3-26 shows the method to dynamically activate the newly defined CPU from the Linux guest.

Example 3-26 Dynamically activating additional CPUs from a z/VM Linux guest

```
[root@lhx001 ~]# echo "1" > /sys/devices/system/cpu/cpu1/online
```

The **echo** command brings the new CPU online. This can be verified by looking at the bottom portion of the **dmesg** command output immediately after the completion of the **echo** command. An entry similar to the one shown Example 3-27 is displayed.

Example 3-27 Portion of dmesg output indicating the CPU activation

```
cpu 1 phys_idx=1 vers=FF ident=0B9D1A machine=2084 unused=8000
```

The same message appears on the console of the z/VM Linux guest system and can also be found in the `/var/log/messages` file.

Note: If the Linux system already has two CPUs and we are activating a third CPU, then the “cpu1” in the absolute path shown in Example 3-26 should be replaced with “cpu2”. If we are activating a fourth CPU, then it becomes “cpu3” and so on.

For the dynamic CPU activation (also known as *CPU hot plugging*) to work from Linux, the `CONFIG_HOTPLUG_CPU` option must be enabled in the running kernel. Not all Linux distributions enable this support by default even though they are running the Linux 2.6 kernel. This can be checked by examining the `/boot/config-xxxx` file on the Linux system; where `xxxx` denotes the full version of the current running kernel and can be found by running the **uname -r** command. If the `CONFIG_HOTPLUG_CPU` option is not enabled in the running kernel, then the kernel must be recompiled with this option enabled to exploit the dynamic CPU activation feature on Linux.

In order to limit the waste of per CPU data structures, in the newest Linux kernel versions, only the CPUs present at the time of IPLing the system can be activated by default. If we dynamically add a new CPU to the virtual machine by following the procedure described in 3.5.1, “Dynamically adding a CPU to a z/VM virtual machine” on page 61, after the IPL of the Linux guest, then it cannot be activated from the Linux guest running the newest kernel versions by default. In such cases, to enable dynamic activation of CPUs, more than what is present at the time of IPL of the Linux guest, the kernel parameter “`additional_cpus=xx`” should be used; where `xx` denotes the number of additional CPUs. For example, if at the time of the IPL of Linux, we have two CPUs defined in the virtual machine, and after the IPL, if we expect to define three more CPUs to the virtual machine, then the value of `xx` should be 3. The `additional_cpus` kernel parameter can be specified in the `/etc/zipl.conf` file as shown in Example 3-28. The entry in blue color shows the kernel parameters definition.

Example 3-28 Specifying the additional_cpus kernel parameter in zipl.conf

```
[ipl]
target = /boot/zipl
image = /boot/image
ramdisk = /boot/initrd
parameters = "root=/dev/dasdb1 selinux=0 TERM=dumb
additional_cpus=4"
```

Note: You must issue the `/sbin/zipl` command for any modification made to `/etc/zipl.conf` to take effect.

3. Verify that the CPUs are added.

The CPU activation can be verified by examining the `/proc/cpuinfo` file. The file contains the information about all the CPUs currently active on the z/VM Linux guest system. Example 3-29 shows a sample `/proc/cpuinfo` file.

Example 3-29 Sample /proc/cpuinfo file of a z/VM Linux guest system

```
[root@lnx001 ~]# cat /proc/cpuinfo
vendor_id       : IBM/S390
# processors    : 2
bogomips per cpu: 2398.61
processor 0: version = FF, identification = 0B9D1A, machine = 2084
processor 1: version = FF, identification = 0B9D1A, machine = 2084
```

3.5.4 Dynamically removing a CPU from a z/VM Linux guest

It is possible to dynamically remove a CPU from a z/VM Linux guest. When a CPU is removed dynamically, all the processes running on that CPU is migrated to the remaining CPUs automatically. It is not possible to disable *all* the CPUs in a z/VM Linux guest. At least one CPU must be active at any point in time for the system to function. Linux handles this automatically and protects the system from crashing.

Note: Linux 2.4 kernels does not support dynamic CPU deactivation.

1. Check the number of CPUs present on the Linux guest.

Examine the `/proc/cpuinfo` file to determine how many CPUs are there currently on the Linux guest system. Example 3-30 shows the content of the `/proc/cpuinfo` file before the CPU removal.

Example 3-30 /proc/cpuinfo file before removing the CPU

```
[root@lnx001 ~]# cat /proc/cpuinfo
vendor_id       : IBM/S390
# processors    : 2
bogomips per cpu: 2398.61
processor 0: version = FF,  identification = 0B9D1A,  machine = 2084
processor 1: version = FF,  identification = 0B9D1A,  machine = 2084
```

Note: If there is only one CPU present, then it cannot be removed.

2. Dynamically remove CPU from a Linux guest running on z/VM.

Removing a CPU dynamically from a Linux guest running on z/VM is very simple and is demonstrated in Example 3-31.

Example 3-31 Removing a CPU dynamically from a Linux guest on z/VM

```
[root@lnx001 ~]# echo "0" > /sys/devices/system/cpu/cpu1/online
```

The corresponding **dmesg** command output is shown in Example 3-32.

Example 3-32 dmesg output after the CPU removal from the Linux guest on z/VM

```
Processor 1 spun down
```

This can be further verified by examining the `/proc/cpuinfo` file. Example 3-33 shows the content of `/proc/cpuinfo` after removing the CPU from the Linux guest on z/VM.

Example 3-33 /proc/cpuinfo file after the CPU removal from the Linux guest on z/VM

```
[root@lnx001 ~]# cat /proc/cpuinfo
vendor_id       : IBM/S390
# processors    : 1
bogomips per cpu: 2398.61
processor 0: version = FF,  identification = 0B9D1A,  machine = 2084
```

Similar to adding a CPU dynamically, dynamic CPU removal can be used only when the `CONFIG_HOTPLUG_CPU` option is enabled in the running Linux kernel. This can be checked by examining the `/boot/config-xxxx` file on the Linux system; where `xxxx` denotes the full version of the current running kernel and can be found by running the **uname -r** command. If the `CONFIG_HOTPLUG_CPU` option is not enabled in the running kernel, then the kernel must be recompiled with this option enabled to exploit the dynamic CPU deactivation feature on Linux.

3.6 Storage management under z/VM

As noted earlier, CP does not virtualize the architectural change notifications that are required to allow dynamic memory changes. Therefore, it is not possible to add or remove the storage allocated to a virtual machine and hence from the guest operating system running in that virtual machine.

It is possible to define a maximum value for the storage allocated to a z/VM virtual machine while creating the directory entry for it. Example 3-34 shows the sample z/VM directory entry for a virtual machine with the maximum storage limit indicated in blue color.

Example 3-34 Sample z/VM directory entry indicating the maximum storage limit

```
USER LNX001 LNX001 512M 1G
INCLUDE LINDFLT
MACHINE ESA 2
MDISK 0191 3390 0001 0080 CMS001 MR
MDISK 0200 3390 0001 3338 LIN001 MR
MDISK 0201 3390 0001 3338 LIN002 MR
```

The z/VM user (that is, virtual machine) “LNX001” shown in Example 3-34 gets 512 MB of storage allocated by default when it is logged in to z/VM. But because the maximum storage for this user is defined as 1 GB, this user can increase the storage up to any value not greater than 1 GB by itself. This can be done using the `DEFINE STORAGE` command without requiring any additional privileges (that is, a Class G user can alter the storage using the `DEFINE STORAGE` command). When the storage is altered using the `DEFINE STORAGE` command, the z/VM virtual machine gets reset. The guest operating system running in the virtual machine is then shut down abnormally, if not shut down gracefully already, as a result of this reset. Therefore, it is very important to bring down the z/VM guest operating system gracefully before altering the storage using the `DEFINE STORAGE` command. Example 3-35 shows the effect of redefining the storage. Initially the virtual machine has 512 MB of storage and the guest operating system running in the virtual machine is CMS (note the `READY` and `CMS`). When the storage is redefined to 1 GB, the z/VM virtual machine storage is cleared and the virtual machine gets reset. As a result of this, the guest operating system running in that virtual machine is destroyed (that is, destroyed from the memory and not from the DASD). The z/VM virtual machine is put back to CP and the z/VM guest operating system must be IPLed again.

```
query storage
STORAGE = 512M
READY; T=0.01/0.01 22:04:01
CMS
define storage 1g
STORAGE = 1G
STORAGE CLEARED - SYSTEM RESET.

CP
ipl cms
Z/VM V5.1.0    2005-11-17 17:55

READY; T=0.01/0.01 22:04:23

query storage
STORAGE = 1G
READY; T=0.01/0.01 22:04:27
```

It is true that this method of storage management is not dynamic. But it gives a flexibility in the sense that, without requiring any special privileges, the z/VM virtual machine user is able to redefine its own storage within the upper limit.

3.7 Resource simulation using z/VM

z/VM does resource simulation to make a virtual machine look like a real machine. This is an attractive feature provided by z/VM because it can be exploited to simulate a device that might not be there on the real physical system. The number of resources that can be simulated on a z/VM system is large. They include CPU, memory, open systems adapter (OSA), fixed-block architecture (FBA) DASD, channel-to-channel (CTC) adapter, printer, reader, punch, console, and so on. We show some examples in this chapter to demonstrate the simulation of devices which are not there on the physical system. The examples that we cover in the remainder of this chapter are simulation of a multiprocessor system and simulation of an FBA DASD device. Further, in Chapter 6, “Parallel Sysplex under z/VM” on page 97, we discuss about running a virtual Parallel Sysplex under z/VM without requiring a real coupling facility and coupling links.

3.7.1 Simulating a multiprocessor system using z/VM

Using z/VM, it is possible to simulate a multiprocessor system on a real system with only one processor (uniprocessor). z/VM does this by creating *virtual processors* on the real processor. These virtual processors are despatched on the real processor by z/VM using the SIE instruction. A virtual machine running on z/VM can have a maximum of 64 virtual processors. In addition, the virtual processors possess all of the characteristics of the real processors on which they are running. Example 3-36 shows a sample z/VM directory entry for a virtual machine, which allows up to four virtual processors to be created by that z/VM virtual machine user.

Example 3-36 Sample z/VM directory for creating 4 virtual CPUs

```
USER LNX001 LNX001 512M 1G
INCLUDE LINDFLT
MACHINE ESA 4
MDISK 0191 3390 0001 0080 CMS001 MR
MDISK 0200 3390 0001 3338 LIN001 MR
MDISK 0201 3390 0001 3338 LIN002 MR
```

The entry highlighted in blue color authorizes the z/VM virtual machine to create up to four virtual processors by its own.

Note: Defining more virtual processors than the number of real processors results in poor performance. This kind of configuration is only recommended for testing.

Referring the sample z/VM directory in Example 3-36, by default, the virtual machine “LNX001” gets only one virtual processor when it is started (that is, logged in to z/VM). Additional processors must be defined using the DEFINE CPU command. It is convenient to place the correct DEFINE CPU statements in the PROFILE EXEC of the virtual machine so that whenever the virtual machine is started (logged in to z/VM), the required number of processors are defined automatically. This avoids the need to type the DEFINE CPU command manually every time the virtual machine is started. Example 3-37 shows a sample PROFILE EXEC, which can be used to get four virtual processors defined automatically every time the virtual machine is started.

Example 3-37 Sample PROFILE EXEC for creating four virtual processors

```
/* SAMPLE PROFILE EXEC TO DEFINE 4 VIRTUAL CPUS */
'CP DEFINE CPU 01'
'CP DEFINE CPU 02'
'CP DEFINE CPU 03'
```

One processor (the BASE processor) is always created for a virtual machine when it is started or created (that is, logged in to z/VM). It is only necessary to define the remaining three processors in the PROFILE EXEC. Also, note that the processors are numbered starting from zero and incremented by one. Hence, the first processor is assigned the numeric value of zero.

It is possible to define all the three processors in a single statement as shown in Example 3-38.

Example 3-38 Alternate sample PROFILE EXEC

```
/* ALTERNATE SAMPLE PROFILE EXEC TO DEFINE 4 VIRTUAL CPUS */  
'CP DEFINE CPU 1-3'
```

It is also possible to define the processors directly as part of the z/VM directory entry for the virtual machine as shown in Example 3-39. The entries highlighted in blue color ensure that the virtual machine gets four virtual processors when logged in and does not need any DEFINE CPU statement in the PROFILE EXEC of that virtual machine.

Example 3-39 Defining the virtual processors as part of the z/VM directory entry

```
USER LNX001 LNX001 512M 1G  
INCLUDE LINDFLT  
MACHINE ESA 4  
CPU 00  
CPU 01  
CPU 02  
CPU 03  
MDISK 0191 3390 0001 0080 CMS001 MR  
MDISK 0200 3390 0001 3338 LIN001 MR  
MDISK 0201 3390 0001 3338 LIN002 MR
```

The QUERY CPUS command can be used from a virtual machine to display the processors defined in it. Example 3-40 shows the output of the QUERY CPUS command.

Example 3-40 QUERY CPUS command output from a z/VM virtual machine

```
query cpus  
00: CPU 00 ID FF0B9D1A20848000 (BASE)  
00: CPU 01 ID FF0B9D1A20848000 STOPPED  
00: CPU 02 ID FF0B9D1A20848000 STOPPED  
00: CPU 03 ID FF0B9D1A20848000 STOPPED  
READY; T=0.01/0.01 01:32:18
```

The STOPPED status in the output means that the virtual machine has not yet started using the these CPUs. When the guest operating system starts using the processors, the STOPPED state changes automatically.

3.7.2 Simulating an FBA DASD device using z/VM

In this section, we demonstrate how to simulate an FBA DASD device using z/VM without requiring a real FBA DASD device on the physical system. This is done using the concept of virtual disks (VDISKS) in storage. A virtual disk in storage is allocated from system storage and not mapped to a real DASD device. Therefore, defining a virtual disk in storage does not require having a real FBA DASD device on the physical system.

Before creating a VDISK, we have to check the maximum amount of system storage that is available for allocation as virtual disks in storage. Example 3-41 shows the way to check this using the **query vdisk syslim** command.

Example 3-41 Checking the system storage available for VDISK creation

```
query vdisk syslim
VDISK SYSTEM LIMIT IS      1202016 BLK,          0 BLK IN USE
Ready; T=0.01/0.01 00:29:07
```

The command output shows the number of 512-byte blocks available for VDISK use. It is possible to set a limit on the maximum resource available for virtual disks in storage created by a single user by using the **define** command. Example 3-42 shows the way to check this using the **query vdisk userlim** command.

Example 3-42 Checking the user limit for VDISK creation

```
query vdisk userlim
VDISK USER  LIMIT IS      144000 BLK
Ready; T=0.01/0.01 00:35:34
```

Note: The user limit does not apply to virtual disks in storage defined by MDISK statements in the z/VM user directory.

The **set vdisk** command (which requires privilege class B) can be used to alter the system and user limits.

The **define** command is used to create a virtual disk in storage. Example 3-43 shows the way to use the **define** command to create a VDISK in storage.

Example 3-43 Using the define command to create a VDISK in storage

```
define vfb-512 as 229 204800
DASD 0229 DEFINED
query virtual 229
DASD 0229 9336 (VDSK) R/W      204800 BLK ON DASD  VDSK SUBCHANNEL =
0035
```

The **define vfb-512** command means that a fixed block (512-byte blocks) VDISK should be created. 229 is an arbitrary virtual device number for the VDISK to be created. This number can be any thing, but it should not already be in use in the same z/VM virtual machine. 204800 means that the new VDISK should be created with 204800 blocks of 512 bytes (that is, 1 GB). The **query virtual 229** command output indicates that the VDISK is created as an FBA (9336) and ready for use.

A VDISK is destroyed when the last user linked to it detaches the link or logs off from the system. Therefore, VDISKs are good candidates for transient data storage only. Examples of such usage are testing, use as a swap space, and so on.

Note: Swap space created using a VDISK is faster than a swap space defined on a DASD.

The VDISK can also be created by adding the appropriate z/VM directory statement to the directory entry for a virtual machine. A sample z/VM directory statement is shown in Example 3-44, which is an alternative way to create the VDISK shown in Example 3-43.

Example 3-44 Sample z/VM directory statement for creating a VDISK

```
MDISK 0229 FB-512 V-DISK 204800
```

When a VDISK is defined as part of the directory entry, it is created when the first user links to it and is destroyed when the last user detaches it or logs off from z/VM.

3.8 References

Refer to the following for more information about the topics discussed in this chapter.

- ▶ *z/VM V5R2.0 General Information*, GC24-6095-04
- ▶ *z/VM V5R2.0 System Operation*, SC24-6121-01
- ▶ *z/VM V5R2.0 Running Guest Operating Systems*, SC24-6115-01
- ▶ *z/VM V5R2.0 CMS Commands and Utilities Reference*, SC24-6073-01
- ▶ *z/VM V5R2.0 CP Planning and Administration*, SC24-6083-03
- ▶ *z/VM V5R2.0 CP Commands and Utilities Reference*, SC24-6081-03
- ▶ *Linux for IBM System z9 and IBM zSeries*, SG24-6694
- ▶ (Linux Kernel 2.6) *Linux on System z - Device Drivers, Features, and Commands*, SC33-8289-02
- ▶ (Linux Kernel 2.4) *Linux on zSeries - Device Drivers and Installation Commands*, SC33-8282

Outlook on z/VM Version 5 Release 3.0

This chapter describes some of the new functions, features, and enhancements that will be introduced with z/VM Version 5 Release 3.0, which is scheduled for general availability (GA) in June 2007. We do not describe all of the new features and enhancements. Instead, we choose the items that we think are interesting for the use of guest operating systems.

In this chapter, we discuss the following topics:

- ▶ General consideration
- ▶ Processor and device support
- ▶ Networking
- ▶ Virtualization
- ▶ Systems management
- ▶ Security
- ▶ Guest American Standard Code for Information Interchange (ASCII) console support

4.1 Improvements in z/VM 5.3

We describe here on a very high level some of the new features and improvements that came with the z/VM 5.3, and how the guest operating systems can exploit it. Because we did not have access to a z/VM 5.3 installation at the time of writing, we did not prove and test this new functionality.

4.1.1 General outlook

z/VM 5.3 extends mainframe scalability support for hosted virtual servers. This includes even more real-memory constraint relief, support for more than 24 CPUs in a single z/VM image, improved disk input/output (I/O) throughput with z/VM exploitation of HiperPAV and enhanced real network I/O throughput with Link Aggregation support.

z/VM 5.3 provides additional z/VM exploitation of zSeries processor, I/O and networking technology, while advancing the state of the art in virtualization support for guest images. z/VM 5.3 only supports z/Architecture-capable (64-bit) processors. Specific processors supported include: z800, z900, z890, z990, z9-109 and any new System z9 servers offered prior to the general availability of z/VM 5.3.

Like z/VM 5.2, the z/VM 5.3 Control Program (CP) will not IPL in ESA/390 mode. Users will still be able to create ESA/390 and z/Architecture mode virtual machines on the same copy of the z/VM Control Program. This is a valuable feature for those who want to run a mix of 31-bit/32-bit images with 64-bit systems.

4.1.2 Processor and device support

z/VM Version 5 Release 3.0 will provide support for customer use by allowing up to 32 CPUs within a single image of z/VM. This support allows clients to increase the CPU capacity for their z/VM-hosted workloads, complementing the enhanced real-memory scalability support introduced in z/VM 5.2 and extended with this release.

Support has been added to z/VM 5.3 to allow virtual machines to be configured with virtual zSeries Application Assist Processors (zAAPs). This will allow z/OS guest images to dispatch zAAP-capable workload on virtual zAAPs. Real standard engines (CPs) will be used by the z/VM Control Program to deploy virtual zAAP instructions. This function will help z/OS customers assess the potential value of adding zAAP engines to their production z/OS environment as well as exercise zAAP code path in z/OS.

Support has been added to z/VM 5.3 to improve IBM ECKD™ FlashCopy® Version 2 operations in a z/VM environment. Functional and usability enhancements in this area will facilitate the automation of backup operations in z/VM.

4.1.3 Networking

Link aggregation support allows OSA-Express devices to be grouped in a manner that allows near seamless failover for Layer 2 switches and the potential for increased network bandwidth for virtual switch traffic. The performance capabilities of z/VM guest LAN support is improved with z/VM 5.3 by exploiting 64-bit instructions and Access Register mode.

4.1.4 Virtualization

Server virtualization enhancements introduced with z/VM 5.3 include additional support for large real memory configurations. The z/VM 5.3 Control Program will be able to store page management control blocks above the 2 GB address line. This will enable customers to achieve improved system throughput for memory-intensive environments configured with 256 GB or more of real memory. One of z/VM distinctions in the marketplace is the ability to host large virtual server environments on a single z/VM image. This capability is extended with z/VM 5.3 and further facilitates the ability to exploit the advantages of hosting virtual servers on a single hypervisor, advantages such as shared-memory exploitation and operational ease of management.

The Virtual Switch SNMP agent in this release will enable solution providers to access performance and operational data pertaining to a z/VM virtual switch. This will help network administrators plan network growth, solve network problems, and manage network performance.

4.1.5 Systems management

z/VM support for systems management is improved with this release. The z/VM system management application programming interfaces (APIs) are being restructured to use a socket-based server interface. This will simplify the task of adding new APIs as well as enable multi-tasking capabilities. New APIs are being added as well. A key exploiter of the APIs is the IBM Director product, now available with z/VM 5.2 and 5.1.

The z/OS V1.8 Lightweight Directory Access Protocol (LDAP) server was ported to z/VM 5.3. This will allow customers to deploy a common user name space between z/VM and Linux. This support for a common user name space requires the use of the Resource Access Control Facility (RACF) for VM feature.

4.1.6 Security

Until now the passwords in z/VM were limited to be between 1 - 8 characters, alphanumeric and uppercase. These limitations can prevent z/VM users from conforming to the security policies of organizations that require passwords of more than 8 characters. In z/VM 5.3, the support has been added to use longer passwords and password phrases in order to conform to the more modern password security requirements.

- ▶ A long password is a single word of arbitrary length up to 200 characters.
- ▶ A password phrase (or passphrase) is:
 - A mixed case string of arbitrary length but longer than the longest password.
 - Any character:
 - Slash and quotes are permitted but not recommended.
 - Blanks are allowed, leading and trailing blanks are stripped.
 - The external security manager (ESM) may restrict the use of certain characters.
 - Single quotation marks must surround the passphrase if required by context.
 - Where blanks, quotation marks, or slashes are ambiguous
 - Characters outside of quotation marks not permitted

To use these new long passwords and password phrases, an ESM is required. The system gives the password as is to the ESM for authentication. If no ESM is installed or does not support password phrases, then any password longer than 8 characters is not valid.

The password phrases are supported within z/VM by:

- ▶ Logon
- ▶ TCP/IP
- ▶ Performance toolkit
- ▶ System management API

What is not changing:

- ▶ CP Directory: The user directory statement does not support long passwords or passphrases.
- ▶ **AUTOLOG** and **XAUTOLOG** will not allow password phrases.
- ▶ Minidisk password remains 1 - 8 uppercase characters.

- ▶ APPC/LU6.2 applications and infrastructure: The architecture limits passwords to 10 characters, therefore there is no significant benefit in modifying this support for:
 - TSAF
 - AFFCVM
 - CPI communication
 - AVS
 - ISFC

In order to support the longer passwords and passphrases, the password field on the logon screen has been extended.

4.1.7 Guest ASCII console support

In the z/VM 5.3, z/VM guest support has been added for the integrated ASCII console. This support allows a Linux system running as a guest under z/VM to use the ASCII console provided with the Hardware Management Console (HMC) as a full screen system console.

This support is primarily intended to provide an emulated VT220 console to the Linux guest system when other network-based console options are not available.

The integrated ASCII console provides an ASCII console session from the HMC, which can be used by a Linux system running in a logical partition (LPAR) or as a guest of z/VM. To use the console, it must be opened for the LPAR of the z/VM or Linux system in which it will be used.

When can it be useful to use the integrated ASCII console: Use this device when your network connection is down and you have to make emergency fixes to your Linux guest. Otherwise, you will be forced to use line mode tools such as the **ed** editor to make repairs from the guest z/VM line mode console.

System recovery is considered the primary use of this device, but it can be used at any other time it is convenient to do so.

How to use it

To use the integrated ASCII console:

1. Configure your Linux guest in advance to use the ASCII console.
2. Attach the console to the guest virtual machine using the **attach sysascii** command.
3. Use the integrated ASCII console to run full screen commands and tools such as **vi** to repair the Linux guest. Linux can use the ASCII console like a full screen console, but it is just using its scrolling line mode.



Part 2

Developing, deploying, and managing your environment

In this part, we describe some of the practical uses of z/VM such as:

- ▶ The Sysplex Trainer
- ▶ z/TPF
- ▶ Dynamic resource management
- ▶ Virtualization

Cloning a production environment (a practical scenario)

This chapter discusses why you might want to create a test version of your production Linux environment under z/VM and how to go about doing so. We highlight many of the benefits of z/VM and also describe several best practices to consider when planning the environment. Based on the procedures described in this chapter, other operating systems such as z/OS can be duplicated and used under z/VM. If z/OS is of particular interest to you, be sure to read Chapter 6, “Parallel Sysplex under z/VM” on page 97, where we branch out to cover how your z/OS Parallel Sysplex environment can be used and tested under z/VM.

5.1 Introducing the sample environment

A typical production environment might consist of one to many systems running IBM WebSphere® Portal Server in a cluster, one or more clustered IBM DB2® instances, and an HTTP sprayer to distribute requests across the WebSphere Portal Server cluster, as shown in Figure 5-1.

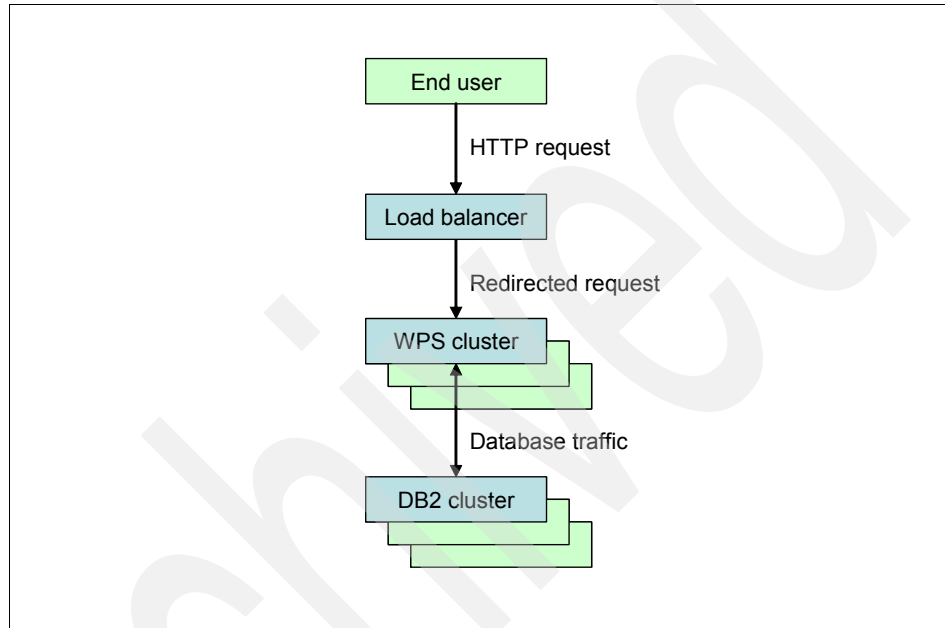


Figure 5-1 Sample production environment

In this chapter, we demonstrate how to clone a scaled-down version of this environment running in *logical partition* (LPAR) mode to an identical version running under z/VM. Our sample environment consists of three Linux systems: A WebSphere Portal Server system, a DB2 system, and a third running the Load Balancer component of *WebSphere Application Server Edge Components*. In Figure 5-1, these systems are shown in light blue.

5.2 Best practices for replicating environments

There are a number of considerations that should be kept in mind when planning a production and accompanying test environment. If properly executed, replicating and managing an environment can be very easy.

5.2.1 Determining storage and paging allocation

One of the primary advantages of using z/VM is that it allows you to overcommit the storage that is available to the LPAR it is running in. This means that you should be able to run all of the same systems that you have in your production environment with less storage in your test environment.

Many z/VM experts recommend keeping the overcommit ratio below 2:1 or even 1.5:1. That means that if your system needs 1000 MB of storage, z/VM only needs about 666 MB of real storage to back it (using the more conservative 1.5:1 ratio).

Note: If your workloads continually vie for storage, the overhead of having z/VM page might reduce the system performance.

In terms of paging space, you should have, at a minimum, as much as the total virtual storage your systems will be using. If you have additional direct access storage device (DASD) available, having as much as two times the amount of virtual storage is recommended so that at most only 50% of your page space is in use.

One forthcoming technology that will improve z/VM's ability to overcommit storage is *collaborative memory management* (CMM), stage 2. CMM2 allows guest operating systems to tell the hypervisor which memory pages they are actually using, which allows z/VM to steal unused pages and give them to another guest in need. Patches that enable this feature in the Linux kernel have already been released and should be incorporated into Linux distributions in the near future.

5.2.2 Allocating DASD

There are four things that you should keep in mind when determining what DASD volumes to allocate your z/VM systems and the guests running in it:

- ▶ Select volumes from different ranks in the storage subsystem.
- ▶ Try to have multiple channel paths to the DASD.
- ▶ If possible, use a storage subsystem that offers FlashCopy V2.
- ▶ When available, use parallel access volumes (PAVs) if the guest operating system supports it.

We explain each of these points in a little more depth.

Select volumes from different ranks

I/O performance can benefit greatly if DASD volumes are selected from different ranks in the storage subsystem, even when using a storage subsystem as advanced as the IBM TotalStorage® DS8000™. More information about configuring and selecting volumes from a storage subsystem can be found in *The IBM TotalStorage DS8000 Series: Concepts and Architecture*, SG24-6452, particularly in Section 5.3, “The abstraction layers for disk virtualization.”

Use multiple channel paths

One other important consideration when allocating DASD is the availability of paths to the DASD. Having multiple IBM ESCON® or IBM FICON® channels to a storage subsystem helps prevent input/output (I/O) bottlenecks as individual channels are saturated.

Make sure FlashCopy is available

IBM storage products offer a feature called FlashCopy, which enables you to make point-in-time, full volume copies of data, with the copies immediately available for read or write access. In order to take advantage of this feature, try to allocate the DASD volumes for your systems from the same storage subsystem whenever possible. This will allow you to copy volumes almost instantaneously, a significant improvement over the traditional *DASD Dump Restore* (DDR) method.

Use parallel access volumes if the guest supports it

The z/Architecture only allows for a single I/O operation to be processed for a logical volume at any one time. In order to work around this, IBM introduced parallel access volumes (PAV) in its storage products. PAV lets a storage administrator define a base volume and then create additional aliases for that volume. This allows a guest that can exploit it to simultaneously process multiple I/O operations to the same logical volume. Currently both Linux on System z and z/OS support PAV.

Note: z/VM itself cannot take advantage of PAV for system volumes (such as the resource, paging, and spool volumes). This capability will likely be added to a future version of z/VM.

If you do decide to take advantage of PAV, it is often easier to set up when the storage subsystem is initially configured so device addresses can be reserved for PAV aliases.

More information about PAV can be found in *Linux on System z - How to Improve Performance with PAV*, SC33-8292, and also on the z/VM Web site:

<http://www.vm.ibm.com/storman/pav/pav2.html>

5.2.3 Managing the CP directory

The Control Program (CP) directory is a flat file that is used to manage guest definitions for z/VM. Keeping track of the information contained in it can quickly grow difficult if you have more than a few z/VM guests, but using a directory manager such as IBM Directory Maintenance Facility (DirMaint) can vastly simplify the administration of users and DASD in a z/VM environment. While DirMaint is a priced feature of z/VM, investing in it early on can prevent many system administration headaches in the future. DirMaint is also used by Director, an IBM primary system management product, which we discuss in Chapter 9, “z/VM system management tools” on page 123. For our examples in this chapter, we rely on DirMaint to manage the directory for us.

Refer to *zSeries Platform Test Report for z/OS and Linux Virtual Servers*, SA22-7997-02, for examples of scripts that can be used to clone z/VM guests. More information about cloning IBM middleware, including WebSphere Application Server, DB2, and MQ Series, can be found in *z/VM and Linux on IBM System z: The Virtualization Cookbook Version 2* and *WebSphere Portal Server for Linux on zSeries and z9*, REDP-4175.

5.3 Planning considerations

In order to replicate an existing environment, you have to know what devices are being used by each system in the environment so that you can define the z/VM guests for the new test systems to match the device configuration. This allows you to bring up the systems with few to no operating system configuration changes.

Note: Changing device addresses for a system (especially a running system) can be quite difficult and should be avoided if at all possible.

To start, let us examine which DASD and network addresses the existing systems are already using. You will want to record the list so that you have it readily available when you define the new z/VM guests. The process for determining which devices are in use varies from operating system to operating system.

5.3.1 Determining device usage in Linux

The easiest way to determine what devices are being used by your System z Linux guest is to use the `lsdasd`, `lsqeth`, and `lstape` commands provided by the

s390-tools package IBM develops. These three programs will print device information for any online devices of that type, as show in Example 5-1.

Note: The `lscss` command can be used to display all devices that are available to your Linux guest, whether or not they are varied online in Linux.

Example 5-1 Sample output from `lsdasd`, `lsqeth`, and `lstape`

```
# lsdasd
0.0.0F00(ECKD) at ( 94: 0) is dasda      : active at blocksize 4096,
601020 blocks, 2347 MB
0.0.0F01(ECKD) at ( 94: 4) is dasdb      : active at blocksize 4096,
601020 blocks, 2347 MB
0.0.0F02(ECKD) at ( 94: 8) is dasdc      : active at blocksize 4096,
601020 blocks, 2347 MB
# lsqeth
Device name                : eth0
-----
      card_type             : OSD_1000
      cdev0                  : 0.0.0600
      cdev1                  : 0.0.0601
      cdev2                  : 0.0.0602
      ...
# lstape --online
TapeNo  BusID      CuType/Model DevType/Model  BlkSize State  Op
MedState
0        0.0.0530  3590/60      3590/11        auto   UNUSED  ---
UNLOADED
```

5.3.2 Determining device usage in z/VM

As in Linux, there are three commands available to display available devices: QUERY VIRTUAL DASD, QUERY VIRTUAL NIC, and QUERY VIRTUAL TAPES. Example 5-2 shows sample output from these commands.

Example 5-2 Sample output from `query virtual dasd`, `query virtual nic`, and `query virtual tapes`

```
query virtual dasd
DASD 0F00 3390 0X0F00 R/W      3339 CYL ON DASD  0F00 SUBCHANNEL =
0007
DASD 0F01 3390 0X0F01 R/W      3339 CYL ON DASD  0F01 SUBCHANNEL =
0008
DASD 0F02 3390 0X0F02 R/W      3339 CYL ON DASD  0F02 SUBCHANNEL =
0009
```

```

Ready; T=0.01/0.01 11:06:12
query virtual nic
Adapter 0600 Type: QDIO      Name: UNASSIGNED  Devices: 3
  Port 0 MAC: 02-00-00-00-00-42  VSWITCH: SYSTEM VSW1
Ready; T=0.01/0.01 11:07:06
query virtual tapes
TAPE 0530 ON DEV  0530 3590 R/W SUBCHANNEL = 0013
TAPE 0531 ON DEV  0531 3590 R/W SUBCHANNEL = 0014
TAPE 0532 ON DEV  0532 3590 R/W SUBCHANNEL = 0015
TAPE 0533 ON DEV  0533 3590 R/W SUBCHANNEL = 0016
Ready; T=0.01/0.01 11:07:45

```

5.3.3 Determining device usage in z/OS

The following command displays the first 999 online DASD devices that are available to a z/OS system:

```
D U,DASD,ONLINE,000,999
```

A similar command is available to display the online network devices:

```
D U,CTC,ONLINE,0,999
```

Sample output from both of these commands is shown in Example 5-3.

Example 5-3 D U output from z/OS

```

D U,DASD,ONLINE,000,9999
IEE457I 12.01.17 UNIT STATUS 254
UNIT TYPE STATUS      VOLSER      VOLSTATE
D050 3390 0          #@$#L2     PRIV/RSDNT
D051 3390 0          #@$#A1     PRIV/RSDNT
D052 3390 A          #@$#D2     PRIV/RSDNT
...
D U,CTC,ONLINE,000,9999
IEE457I 13.18.52 UNIT STATUS 519
UNIT TYPE STATUS      VOLSER      VOLSTATE
2000 OSA  A-BSY
2001 OSA  A
2002 OSA  A-BSY
2003 OSA  0
...

```

More information about the DISPLAY U command can be found in *MVS System Commands*, SA22-7627.

5.4 Setting up z/VM to mimic your existing environment

Now that you know what DASD, network, and tape devices your production systems are using, it is time to create z/VM guests that have identical device configurations. The following scenario assumes that you are using DirMaint to manage your directory and have no external security manager (ESM), such as IBM Resource Access Control Facility (RACF), enabled.

5.4.1 Creating a virtual network

In order for the test systems to be useful, there must be some form of network for them to communicate with one another. For our sample environment, we recommend creating a VSWITCH, as described in 2.3, “VSWITCH” on page 27, although a QDIO guest LAN is also acceptable.

5.4.2 Adding guests to the CP directory

The first step is to create the z/VM directory entries for each of the three systems that we will be cloning and then have DirMaint add those guests. If you are not using DirMaint, you will have to edit the USER DIRECT file and follow the steps in Chapter 17, “Creating and Updating a User Directory”, in *CP Planning and Administration*, SC24-6083, to make the changes.

1. To begin, log on to your z/VM system as the MAINT user and create the LINWPS01 DIRECT file using XEDIT:

XEDIT LINWPS01 DIRECT
2. Add the statements shown in Example 5-4. If you require assistance with XEDIT, consult the *XEDIT User's Guide*, SC24-6132. Notice that the virtual device addresses (0F00, 0F01, 0F02) are the same as the real device addresses in our production systems, but the real devices (5700, 5800, 5900) can be completely different.

Example 5-4 Contents of LINWPS01 DIRECT

```
USER LINWPS01 23456789 512M 2G G
  INCLUDE IBMDFLT
  IPL CMS PARM AUTOOCR
  MACHINE ESA
  MDISK 0200 FB-512 V-DISK 2048000 MR
  MDISK 0F00 3390 DEVNO 5700 MR ALL ALL ALL
  MDISK 0F01 3390 DEVNO 5800 MR ALL ALL ALL
  MDISK 0F02 3390 DEVNO 5900 MR ALL ALL ALL
```

*

3. File the changes and then use the following command to add the guest to the directory:

```
DIRM ADD LINWPS01
```

You should see an output similar to that shown in Example 5-5.

Example 5-5 DirMaint output when adding a guest

```
DVHXT1191I Your ADD request has been sent for processing.
DVHREQ2288I Your ADD request for LINWPS01 at * has been accepted.
DVHBIU3450I The source for directory entry LINWPS01 has been updated.
DVHBIU3424I The next ONLINE will take place immediately.
DVHDRC3451I The next ONLINE will take place via delta object directory.
DVHBIU3428I Changes made to directory entry LINWPS01 have been placed
DVHBIU3428I online.
DVHREQ2289I Your ADD request for LINWPS01 at * has completed; with RC
DVHREQ2289I = 0.
```

4. LINWPS01 is now available for use. Next, repeat the process of creating a directory file and adding it to the directory for LINDB01 and LINSPRAY, whose contents are shown in Example 5-6 and Example 5-7.

Example 5-6 Contents of LINDB01 DIRECT

```
USER LINWPS01 23456789 512M 2G G
  INCLUDE IBMDFLT
  IPL CMS PARM AUTO CR
  MACHINE ESA
  MDISK 0200 FB-512 V-DISK 2048000 MR
  MDISK 0F03 3390 DEVNO 5701 MR ALL ALL ALL
  MDISK 0F04 3390 DEVNO 5801 MR ALL ALL ALL
  MDISK 0F05 3390 DEVNO 5901 MR ALL ALL ALL
*
```

Example 5-7 Contents of LINSPRAY DIRECT

```
USER LINSPRAY 23456789 256M 1G G
  INCLUDE IBMDFLT
  IPL CMS PARM AUTO CR
  MACHINE ESA
  MDISK 0200 FB-512 V-DISK 2048000 MR
  MDISK 0F06 3390 DEVNO 5702 MR ALL ALL ALL
  MDISK 0F07 3390 DEVNO 5802 MR ALL ALL ALL
  MDISK 0F08 3390 DEVNO 5902 MR ALL ALL ALL
*
```

For more information about the structure of a directory entry, see Chapter 17, “Creating and Updating a User Directory”, in *CP Planning and Administration*, SC24-6083.

While you are logged on as MAINT, you will also want to grant permission for these three guests to access the virtual LAN, assuming that you created a VSWITCH. See 2.3, “VSWITCH” on page 27 for details about how to do this.

5.4.3 Copying DASD volumes

You now have your three z/VM guests defined and it is time to copy the logical volumes from the production systems to the new test systems. The fastest way to do this is to attach the production volumes to the z/VM system and use CP’s FLASHCOPY command (which in turn invokes it on the underlying storage subsystem) to instantly copy the data.

Note: The FLASHCOPY command requires class B privileges, therefore it must be run from MAINT or another privileged user.

Points to consider

There are two potential problems with this approach. The first is that the source and target volumes might be located on different storage subsystems, preventing you from using FlashCopy. If FlashCopy is not available, you can still use the DDR method, but the process of copying data from one volume to the other will take significantly longer (typically 10 minutes to 20 minutes).

The second is that guest operating systems such as Linux use disk buffers and caches quite heavily, which FlashCopy cannot compensate for. What this means is that any uncommitted changes that reside only in memory will not be copied to the target volume. If any kind of critical change was made, the resulting copy might not be bootable or data might be corrupt.

In situations where this is a concern, possible solutions include using the most recent system backups instead of the actual production volumes or waiting until the system is down and maintenance is being performed to perform the copy using FlashCopy.

That said, if you are willing to take a slight risk, it might be appropriate to just flush the file system buffers and then copy the volume using FlashCopy while the system is live. In Linux, there is a command called **sync** that does this. See the **sync** man page (**man sync**) for a few more details.

Based on your own particular situation, you should select the appropriate method. For our sample environment, we assume that all of the DASD is located

in the same storage subsystem and that a few extra minutes can be taken to copy the volumes using FlashCopy while the production system is down for maintenance.

Preparing the volumes to be copied

In Appendix A, “Sample scripts to use when cloning systems” on page 145, you can find several shell scripts that have to be copied to the production volumes. When the production volumes are copied and the resulting systems are IPLed, they will have the same network settings as your existing production systems, which is probably not what you want. The shell scripts from Appendix A, “Sample scripts to use when cloning systems” on page 145, will let you easily change the settings after the volumes have been copied. After you have copied or created each of those scripts to your production Linux systems, you can proceed with copying the volumes.

Using FlashCopy to copy volumes

Using FlashCopy is quite easy. Just attach the source and target volumes to a guest, issue the FLASHCOPY command, and then detach the volumes.

You will want to log on as MAINT and then follow Example 5-8 to successfully copy the volumes. If you prefer to use a pre-written script, see Appendix A, “Sample scripts to use when cloning systems” on page 145.

Example 5-8 Using FlashCopy to clone DASD volumes

```
ATTACH F00-F02 *
F00-F02 ATTACHED TO MAINT
Ready; T=0.01/0.01 14:31:36
LINK LINWPS01 F00 A00 MR
Ready; T=0.01/0.01 14:31:43
LINK LINWPS01 F01 A01 MR
Ready; T=0.01/0.01 14:31:46
LINK LINWPS01 F02 A02 MR
Ready; T=0.01/0.01 14:31:55
FLASHCOPY F00 0 END TO A00 0 END
Command complete: FLASHCOPY F00 0 END TO A00 0 END
Ready; T=0.01/0.01 14:32:24
FLASHCOPY F01 0 END TO A01 0 END
Command complete: FLASHCOPY F01 0 END TO A01 0 END
Ready; T=0.01/0.01 14:32:42
FLASHCOPY F02 0 END TO A02 0 END
Command complete: FLASHCOPY F02 0 END TO A02 0 END
Ready; T=0.01/0.01 14:33:08
```

```
DETACH F00-F02
0F00-F02 DETACHED
Ready; T=0.01/0.01 14:36:44
DETACH A00-A02
0A00-0A02 DETACHED
Ready; T=0.01/0.01 14:36:52
```

All of these steps in Example 5-8 only copied volumes for a single system. You have to repeat the process for the DB2 and Load Balancer volumes. The process is the same as in the preceding example, but you have to substitute the z/VM guest names and the proper device addresses. If you used the REXX script, all three sets of production volumes were copied and you can proceed.

5.4.4 Updating settings for cloned systems

Now that you have made copies of the production volumes, you have to bring up the systems and change their network settings so that they do not conflict with the existing production systems. If the test systems will be connected to an isolated virtual LAN, changing the settings might not be necessary.

Changing the Linux network settings

We have provided a simple script that will allow you to update the network settings when running *Red Hat Enterprise Linux* (RHEL) or *SUSE Linux Enterprise Server* (SLES) on System z. The script can be found in Appendix A, “Sample scripts to use when cloning systems” on page 145.

You have to boot Linux into single user mode, which means that network interfaces and network services should be started. This allows you to change the settings without disrupting the services.

If you are using RHEL, SLES 10, or another distribution that has the interactive boot menu, you should IPL the system as shown in Example 5-9.

Example 5-9 Booting Linux into single user mode on RHEL

```
#cp ip1 f00
zIPL v1.5.3 interactive boot menu

0. default (linux)
1. linux
```

Note: VM users please use '#cp vi vmsg <input>'

Please choose (default will boot in 15 seconds):

```
#cp vi vmsg 0 single
```

```
Booting default (linux)...
```

If you are using a distribution that does not have the boot menu, you have to boot the system normally, log in as root, and then use the **telinit** command as follows to change to runlevel 1 (single user mode):

```
telinit 1
```

When the system starts, it might prompt you for the root password. If prompted, enter the password and press Enter. You will be left at a root shell with all of the file systems mounted in read/write mode. Run the script as shown in Example 5-10 to change the network settings. The output will be slightly different if you are running SLES.

Example 5-10 Updating Linux network settings

```
# ./fix-network.sh linwps01 10.12.1.100
Changed ifcfg-eth0 in /etc/sysconfig/network-scripts,
backing up as old-ifcfg-eth0
Changed /etc/sysconfig/network, backing up as network.old
Changed /etc/hosts, backing up as /etc/hosts.old
```

Changing WebSphere Portal Server settings

When the network settings have been updated, it will also be necessary to update the configuration data for WebSphere Portal Server and DB2 so that they will continue to function. See Appendix A, “Sample scripts to use when cloning systems” on page 145 for a shell script that will update the WebSphere Portal Server configuration and consult *WebSphere Portal Server for Linux on zSeries and z9*, REDP-4175 for more details about cloning WebSphere Portal Server.

When you have updated the WebSphere Portal Server instance information, you still have to edit the configuration file and update the IP address for your DB2 instance.

1. Connect to LINWPS01 using Secure Shell (SSH) and then change directory to `/opt/IBM/WebSphere/PortalServer/config` or wherever you originally installed WebSphere Portal Server to.
2. Make a backup copy of the `wpconfig.properties` file using this command:

```
cp wpconfig.properties wpconfig.properties.orig
```

3. Now edit `wpconfig.properties` with your preferred text editor. You have to find and update the following four properties with the new values: `WpsXDbName`, `WpsDbNode`, `JcrXDbName`, and `JcrXDbNode`.
4. You have to verify the settings using the `WPSconfig.sh` command as shown in Example 5-11, making sure to substitute your DB2 password where appropriate.

Example 5-11 Validating the database connections

```
# ./WPSconfig.sh validate-database-connection-wps
-DDbPassword=password
# ./WPSconfig.sh validate-database-connection-jcr
-DJcrDbPassword=password
# ./WPSconfig.sh validate-database-connection-feedback
-DFeedbackDbPassword=password
# ./WPSconfig.sh validate-database-connection-likeminds
-DLikemindsDbPassword=password
# ./WPSconfig.sh validate-database-connection-wmm
-DWmmDbPassword=password
# ./WPSconfig.sh validate-database-driver
```

Note: If you experience any errors relating to DB2 when going through Example 5-11, try following step 6 from this document:

http://publib.boulder.ibm.com/infocenter/wpdoc/v510/topic/com.ibm.wp.ent.doc/wpf/cfg_db2.html

More information about changing the WebSphere Portal Server database configuration can be found at the following URLs:

- ▶ WebSphere Portal Server Configuration properties reference:
http://publib.boulder.ibm.com/infocenter/wpdoc/v510/index.jsp?topic=/com.ibm.wp.ent.doc/wpf/inst_prep.html
- ▶ Configuring WebSphere Portal for DB2:
http://publib.boulder.ibm.com/infocenter/wpdoc/v510/topic/com.ibm.wp.ent.doc/wpf/cfg_db2.html

Changing the Load Balancer configuration

Editing the dispatcher configuration is quite simple. Connect to the LINSPray system using SSH and then open the default configuration file in your preferred text editor. The configuration file is typically found in the `/opt/ibm/edge/lb/servers/configurations/dispatcher` directory and is called `default.cfg`. You should change all occurrences of the old IP addresses to the new addresses and then save the file.

Changing the DB2 configuration

The changes required for DB2 to function are very minimal. A single line in the node configuration file for each DB2 instance has to be updated with the new host name of the system. The following command should be run when DB2 is *not* running:

```
echo "0 <new hostname> 0" > <instance path>/sql1lib/db2nodes.cfg
```

Where *hostname* should be replaced with the new host name of the system and *instance path* should be replaced with the directory path to a DB2 instance.

Note: The command given above is only for systems that are not participating in a partitioned DB2 cluster. If your production DB2 server was in such a cluster, this command should not be run and the `db2nodes.cfg` file must be edited by hand and any occurrences of the original host name replaced with the new test system host name.

Additional information about editing the node configuration file can be found at this DB2 Information Center topic:

<http://publib.boulder.ibm.com/infocenter/db2luw/v8/topic/com.ibm.db2.udb.doc/start/t0006350.htm>

5.4.5 Testing the changes

The only remaining step is to test the changes. You should restart each of the three Linux guests: LINDB01, LINWPS01, and LINSPRAY, in that order and then try to connect to the Load Balancer using a Web browser. If they are all configured correctly, you should see your portal start in the browser and function properly.

Parallel Sysplex under z/VM

This chapter provides an overview of the steps to set up a Parallel Sysplex on z/VM. We show some detailed procedures to test a Parallel Sysplex for test and training purposes.

In this chapter, we discuss the following topics:

- ▶ Why Parallel Sysplex under z/VM?
- ▶ Parallel Sysplex under z/VM: Planning and steps
- ▶ Procedure to create a CF failure under z/VM
- ▶ Procedure to recover a CF under z/VM
- ▶ Procedure to create a CF connectivity failure under z/VM
- ▶ Procedure to re-establish the CF connectivity under z/VM
- ▶ Failing and recovering a system under z/VM

6.1 Why Parallel Sysplex under z/VM?

There are some very attractive advantages to z/VM that you should consider for running a Parallel Sysplex under z/VM for test, development, and training purposes. The main advantages are:

- ▶ Isolation

z/VM is ideal for providing a testing environment. You can do just about anything you like in a z/OS virtual machine running under z/VM, without the fear that you might impact anything else. You can break operating systems, CF links, and CFs without going near the real hardware or touching the Hardware Management Console (HMC). Given the nature of the things that we do in a test environment, this provides a great degree of comfort, and permits the users of the system the flexibility to do anything they want without the fear that this might impact the production system.

- ▶ Flexibility

One of the really impressive features of z/VM is the self-contained, virtual environment it provides for you. You can break CF links, CFs, and operating systems, all with a z/VM command. As a result, you can set up a training environment for Parallel Sysplex on z/VM and use it from anywhere that you can log on to z/VM. There is no requirement that the training should be done in the computer area. In fact, if you have dial-in facilities, you can conceivably do all of this from home.

- ▶ Reduced resource requirement

When running in the logical partition (LPAR) mode, each operating system image and each CF requires its own dedicated processor storage. However, if you are running under z/VM, the storage of each virtual machine is paged in and out by z/VM as required, allowing you to run with a significantly smaller amount of processor storage. Obviously, the additional processor storage requirement depends on the way the system is used (that is, when and how intensive).

Note: It is recommended that you do *not* run a *production* Parallel Sysplex under z/VM because it does not avoid a single point of failure.

6.2 Parallel Sysplex under z/VM: Planning and steps

This section provides an overview of the planning involved in setting up a Parallel Sysplex under z/VM. Additionally, we provide the steps necessary in order to set up a *virtual* Parallel Sysplex under z/VM.

6.2.1 Planning to set up a Parallel Sysplex under z/VM

To set up a Parallel Sysplex under z/VM, you have the choice of using an existing z/VM system, if there is one already, or use a dedicated z/VM LPAR to run the Parallel Sysplex.

The advantages of setting up a separate z/VM LPAR just for the Parallel Sysplex are that it provides more isolation and a more secure environment. The disadvantages are that it takes more resources than running everything under a single z/VM LPAR and there is one more z/VM system to manage. Select the best method based on your environment and requirement.

There are some hardware and software prerequisites for setting up a Parallel Sysplex under z/VM. Because z/OS requires 64-bit support, it must be done under z/VM 3.1.0 or later. Also, you must be running on an IBM 9672 G5 or later processor. If you want to IPL z/OS 1.7 systems, you must be running on an IBM zSeries or later processor. In addition, because the CFRM couple data set is formatted to support system-managed CF duplexing, the CFs must support CF Level 11 or later.

Note: VM APAR VM63093 is required when using CF Level 12 with a coupling facility virtual machine (CFVM). VM APAR 63958 is required for IPLing a z/OS guest of z/VM 5.2.0 in a virtual machine with multiple CPUs and more than 4 GB of memory.

6.2.2 Steps to set up a Parallel Sysplex under z/VM

To set up a *virtual* Parallel Sysplex under z/VM, you have to perform the following tasks.

1. Plan the sysplex configuration.
2. Create the CF service machines.
3. Create the coupled guest virtual machines.
4. Create the CF service machine console user ID.
5. Update the system directory.
6. Start the CF service machines on z/VM.

7. Establish a coupled connection with the CF service machines.
 - a. Establish a coupled connection manually.
 - b. Establish a coupled connection automatically at logon.
8. Verify the sysplex configuration.
9. IPL all the z/OS images.
10. Define the z/VM CF service machine to z/OS.
11. Establish data sharing among the z/OS virtual machines.
12. Modify the z/OS clock definitions.

A detailed explanation of how to perform the above steps can be found in the manual *z/VM V5R2.0 Running Guest Operating Systems*, SC24-6115-01. Here, we provide an explanation of step 11, Establish data sharing among the z/OS virtual machines.

6.2.3 Establishing data sharing among the z/OS virtual machines in a virtual Parallel Sysplex

Working allegiance simulation must be used when running two or more z/OS guests as part of a sysplex configuration using the z/VM guest coupling simulation support. The WRKALLEG operand on the DASDOPT directory statement or the CP SET WRKALLEG command must be used for any minidisk containing CFRM data sets to maintain cross-system lock integrity (and thereby, data integrity) within the sysplex.

The typical way to achieve direct access storage device (DASD) sharing in a virtual sysplex running under z/VM is to define an arbitrary “master” user as the owner of all the DASD devices, define a z/VM directory profile to LINK all these DASD devices, and use this z/VM directory profile in the definition of each z/OS virtual machine.

Example 6-1 shows the portion of a sample z/VM directory entry for the “master” user, called ZOSMAINT, owning all the DASD devices for the use of z/OS images in the virtual sysplex.

Example 6-1 Portion of a sample z/VM directory entry for “master” user

```
*****
* Owner of Shared Disks                                     *
*****
USER ZOSMAINT NOLOG 6M 12M G
  INCLUDE IBMDFLT
  IPL CMS
  MDISK E400 3390 DEVNO E400 MWV
  DASDOPT WRKALLEG
  MDISK E401 3390 DEVNO E401 MWV
```

```

DASDOPT WRKALLEG
MDISK E402 3390 DEVNO E402 MWV
MDISK E403 3390 DEVNO E403 MWV
MDISK E404 3390 DEVNO E404 MWV
...

```

In Example 6-1, we define an arbitrary “master” user called ZOSMAINT as the owner of all the DASD devices used by the z/OS images in the virtual sysplex. The DASDOPT WRKALLEG statement immediately following the full-pack minidisks E400 and E401 means that work allegiance is active on these minidisks. The MWV operand in the MDISK statements tells CP to use its virtual reserve/release with write support in the I/O operations for these full-pack minidisks.

Example 6-2 shows a sample z/VM directory profile with the required LINK statements to access the DASD devices owned by the “master” user explained in Example 6-1.

Example 6-2 Sample z/VM directory profile for z/OS systems

```

*****
* Profile for z/OS systems *
*****
PROFILE ZOSGUEST
MACHINE ESA 15
STORAGE 3G
MAXSTORAGE 10G
CLASS G
LOGONBY ZOSPROG ZOSADMIN ZOSOPER VMADMIN
IPL CMS
STDEVOPT DASDSYS DATAMOVER LIBRARY CTL
OPTION CFUSER TODENABLE SVC76VM APPLMON
SHARE REL 100
SPECIAL 0400 MSGP CF1
SPECIAL E000 3270
SPOOL 000C 2540 READER *
SPOOL 000D 2540 PUNCH A
SPOOL 000E 1403 A
CONSOLE E010 3215 T
DEDICATE 0690 0690 MULTIUSER
LINK MAINT 0190 0190 RR
LINK MAINT 019D 019D RR
LINK MAINT 019E 019E RR
LINK ZOSMAINT 0191 0191 RR
LINK ZOSMAINT E400 E400 MW

```

```
LINK ZOSMAINT E401 E401 MW
LINK ZOSMAINT E402 E402 MW
LINK ZOSMAINT E403 E403 MW
LINK ZOSMAINT E404 E404 MW
...
```

Note: The MULTIUSER operand in the DEDICATE statement attaches a real tape device to each z/OS virtual machine to be serially shared with other z/OS images in the virtual sysplex.

Example 6-3 shows a sample z/VM directory entry for a z/OS virtual machine called as the CFUSER1. Note that the ZOSGUEST profile INCLUDED in the virtual machine definition.

Example 6-3 Sample z/VM directory entry for a z/OS virtual machine

```
USER CFUSER1 LBYONLY
INCLUDE ZOSGUEST
MDISK 0191 3390 0001 0010 MVS191 MR
```

6.3 Procedure to create a CF failure under z/VM

In a virtual Parallel Sysplex running under z/VM, the two CFs actually run in a special type of virtual machine, called a coupling facility virtual machine (CFVM). There will be one CFVM for each CF. In order to create a CF failure situation, you simply issue the z/VM FORCE command to force off one of the CFVMs. This is effectively the same as turning the power off on a stand-alone CF.

To get the name of the CFVM that you want to force, issue the D CF command as shown in Example 6-4. The user ID of the CFVM is contained in the serial number field labelled **A**. Also, take note of the DEVICE NUMBERS, labelled **C**, of the links connecting the CFVM to the CF (you should actually issue the D CF command on *every* system, and take a note of the DEVICE NUMBERS on each system, just in case they are different). This information is required to restore the CF later.

Example 6-4 Finding the name of the CFVM

```
D CF
COUPLING FACILITY SIMDEV.IBM.EN.00000000CFCC1 A
PARTITION: 0 CPCID: 00
CONTROL UNIT ID: 03C0

NAMED FACIL01
```

COUPLING FACILITY SPACE UTILIZATION

...

SENDER PATH	PHYSICAL	LOGICAL	CHANNEL TYPE
80	ONLINE	ONLINE	CFS
C0	ONLINE	ONLINE	CFS

COUPLING FACILITY DEVICE	SUBCHANNEL	STATUS
4030 C	000E	OPERATIONAL/IN USE
4031	000F	OPERATIONAL/IN USE
4032	0010	OPERATIONAL/IN USE
4033	0011	OPERATIONAL/IN USE

...

COUPLING FACILITY SIMDEV.IBM.EN.0000000**CFCC2** **A**

NAMED FACIL02

COUPLING FACILITY SPACE UTILIZATION

...

SENDER PATH	PHYSICAL	LOGICAL	CHANNEL TYPE
81	ONLINE	ONLINE	CFS
C1	ONLINE	ONLINE	CFS

COUPLING FACILITY DEVICE	SUBCHANNEL	STATUS
5030 C	0012	OPERATIONAL/IN USE
5031	0013	OPERATIONAL/IN USE
5032	0014	OPERATIONAL/IN USE
5033	0015	OPERATIONAL/IN USE

Having determined which CF to stop, press the PA1 key (Esc on the PC keyboard) on the z/OS console. This puts the console into CP READ mode, from which we can enter z/VM commands. To fail the CFVM, enter the command:

```
#CP FORCE cfvmuserid
```

Where *cfvmuserid* is the user ID associated with the CFCC that you want to fail.

Note: CLASS A authority is required to enter the FORCE command (for example, the OPERATOR ID has the CLASS A authority).

As soon as the FORCE command is issued, messages similar to those shown in Example 6-5 are displayed on the consoles of all the systems in the virtual sysplex indicating that they have lost connectivity to that CF.

Example 6-5 Messages following a CF failure

```
*IXL158I PATH C0 IS NOW NOT-OPERATIONAL TO CUID: 03C0 700
      COUPLING FACILITY SIMDEV.IBM.EN.0000000CFCC1
      PARTITION: 0 CPCID: 00
*IXL158I PATH 80 IS NOW NOT-OPERATIONAL TO CUID: 03C0 701
      COUPLING FACILITY SIMDEV.IBM.EN.0000000CFCC1
      PARTITION: 0 CPCID: 00
IXC518I SYSTEM #@$1 NOT USING702
      COUPLING FACILITY SIMDEV.IBM.EN.0000000CFCC1
      PARTITION: 0 CPCID: 00
      NAMED FACIL01
      REASON: CONNECTIVITY LOST.
      REASON FLAG: 13300001.
```

The messages are the same as the ones that are seen in case of a single system losing connectivity. As far as each system is concerned, it cannot tell the difference between a broken CF link and a broken CF; in both the cases, z/OS just knows that it can no longer communicate with the CF. It is up to XCF talking with each of the systems to determine if every system has lost connectivity, in which case it is a CF failure, or if just a subset of systems has lost connectivity, in which case it is just a connectivity failure.

Note: The #@\$1 in Example 6-5 is the z/OS system name and is selected purposefully to avoid duplicate name issues with production systems, because it is very unlikely that any one will use these national language characters in their naming conventions.

6.4 Procedure to recover a CF under z/VM

To reinstate the failed CF, we have to restart the associated CFVM and re-establish connectivity to each of the systems. To start the CFVM that we forced off, issue the XAUTOLOG command from an authorized user ID (for example, OPERATOR). To autolog the CFVM, use the command:

```
XAUTOLOG cfvmuserid
```

Where *cfvmuserid* is the user ID of the CFVM that you want logged on.

The next step is to re-establish connectivity between all the systems and the newly restarted CF. The method to do this is basically the same as restarting a failed CF connection from any system; the difference being that we must do this from *every* system in the sysplex. To issue a z/VM command from each system, log on to the z/OS console for each of the systems. Remember that to issue a

VM command, you must press PA1 (Esc on the PC keyboard) on the z/OS console to get into CP READ mode. After giving the CFVM a few minutes to initialize, issue the following command on each of the systems in the virtual sysplex:

```
#CP DEF MSGP cfvmuserid VDEV nnnn
```

Where *cfvmuserid* is the user ID of the CFVM that has just been restarted, and *nnnn* is the device number of the first CF link to that CFVM from that z/OS system. Make sure that you use the correct device number when issuing the DEF MSGP command on each system. Refer to 6.3, “Procedure to create a CF failure under z/VM” on page 102, for the procedure to find the device number. After issuing the DEF MSGP command, messages similar to those in Example 6-6 are shown indicating that the CF is once again connected to the system.

Example 6-6 Messaging indicating CF is available again

```
IXL157I PATH C0 IS NOW OPERATIONAL TO CUID: 03C0 995
      COUPLING FACILITY SIMDEV.IBM.EN.0000000CFCC1
      PARTITION: 0 CPCID: 00
IXL157I PATH 80 IS NOW OPERATIONAL TO CUID: 03C0 996
      COUPLING FACILITY SIMDEV.IBM.EN.0000000CFCC1
      PARTITION: 0 CPCID: 00
IXC517I SYSTEM #0$1 ABLE TO USE:
      COUPLING FACILITY SIMDEV.IBM.EN.0000000CFCC1
      PARTITION: 0 CPCID: 00
      NAMED FACIL01
```

When the CF is available to *all* systems, we should move all the structures that normally reside there back by using the command:

```
SETXCF START,REBUILD,POPCF=cfname
```

6.5 Procedure to create a CF connectivity failure under z/VM

In a virtual sysplex running under z/VM, a facility called *message processor* is used by z/VM to simulate the CF links. There is one message processor for each operating system/CF pair. To create a complete loss of connectivity from one system to one of the CFs, we issue the CP DETACH MSGPROC command from z/VM specifying the z/VM user ID of the CF that we want to lose connectivity to. To find which VM user ID corresponds to the CF that we want to lose connectivity to, issue the D CF command from z/OS. The user ID of the CFVM for each CF is

shown in the serial number field for each CF labeled **A** and **B**, as shown in Example 6-7. The device numbers (**C** and **D**) listed at the bottom half of the output are required when we restore the connectivity to the CF, therefore we should take a note of those now.

Example 6-7 Using D CF command to obtain the CFVM user ID

```
D CF
RESPONSE=#@$1
IXL150I 23.44.45 DISPLAY CF 442
COUPLING FACILITY SIMDEV.IBM.EN.0000000CFCC1 A
PARTITION: 0 CPCID: 00
CONTROL UNIT ID: 03C0

NAMED FACIL01
...
SENDER PATH      PHYSICAL      LOGICAL      CHANNEL TYPE
      80          ONLINE      ONLINE      CFS
      C0          ONLINE      ONLINE      CFS

COUPLING FACILITY DEVICE SUBCHANNEL STATUS
                  4030 C      000E      OPERATIONAL/IN USE
                  4031 C      000F      OPERATIONAL/IN USE
                  4032 C      0010      OPERATIONAL/IN USE
                  4033 C      0011      OPERATIONAL/IN USE

...

COUPLING FACILITY SIMDEV.IBM.EN.0000000CFCC2 B
PARTITION: 0 CPCID: 00
CONTROL UNIT ID: 03C1

NAMED FACIL02
...
SENDER PATH      PHYSICAL      LOGICAL      CHANNEL TYPE
      81          ONLINE      ONLINE      CFS
      C1          ONLINE      ONLINE      CFS

COUPLING FACILITY DEVICE SUBCHANNEL STATUS
                  5030 D      0012      OPERATIONAL/IN USE
                  5031 D      0013      OPERATIONAL/IN USE
                  5032 D      0014      OPERATIONAL/IN USE
                  5033 D      0015      OPERATIONAL/IN USE
```

In this example, if you want to cause a loss of connectivity between the z/OS that this command was issued from, and the CF with a user ID of CFCC1, you press the PA1 key (Esc key on a PC keyboard) on the system console of the z/OS system to go to the CP READ. Then issue the command:

```
#CP DETACH MSGPROC CFCC1
```

After the command is issued, the message shown in Example 6-8 is displayed on the console.

Example 6-8 Message indicating that the CF link is destroyed

```
MSGPROC CFCC1 DETACHED
```

The word “MORE...” is displayed at the bottom right of the screen. Clear the screen to get back to the z/OS system console. Immediately messages similar to those shown in Example 6-9 are displayed indicating that connectivity to the CF has been lost.

Example 6-9 Message following a connectivity failure

```
*IXL158I PATH C0 IS NOW NOT-OPERATIONAL TO CUID: 03C0 700
      COUPLING FACILITY SIMDEV.IBM.EN.0000000CFCC1
      PARTITION:0 CPCID: 00
*IXL158I PATH 80 IS NOW NOT-OPERATIONAL TO CUID: 03C0 701
      COUPLING FACILITY SIMDEV.IBM.EN.0000000CFCC1
      PARTITION:0 CPCID: 00
IXC518I SYSTEM #@$1 NOT USING 702
      COUPLING FACILITY SIMDEV.IBM.EN.0000000CFCC1
      PARTITION: 0 CPCID: 00
      NAMED FACIL01
      REASON: CONNECTIVITY LOST.
      REASON FLAG: 13300001.
```

6.6 Procedure to re-establish the CF connectivity under z/VM

In this case, we have to redefine the message processor that we deleted when we broke the CF connectivity. To do this, we must know the virtual machine name of the CF and the device numbers that are used to communicate with the CF. You should have noted both of these bits of information at the beginning of the test described in 6.5, “Procedure to create a CF connectivity failure under z/VM” on page 105. To re-establish connectivity, log on to the virtual machine running the z/OS guest that lost connectivity. When you are logged in, press PA1

(Esc on the PC keyboard) to get into CP READ mode. Then type the following command:

```
#CP DEF MSGP cfmuserid VDEV nnnn
```

Where *cfmuserid* is the user ID of the CFVM that you want to reconnect to, and *nnnn* is the first device number listed in the output from the D CF command that we issued before removing connectivity. When the DEFINE MSGP command is issued, the z/OS virtual machine re-establishes connectivity with the CFVM, and z/OS automatically makes those links available again. We do not have to issue any z/OS commands to activate the link. To be sure that the paths to the CF are operational again, issue the command:

```
D CF,CFNAME=FACIL01
```

The response displayed in Example 6-10 shows that the two paths, **A** and the devices **B**, are all operational again.

Example 6-10 Checking the CF connectivity

D CF,CFNM=FACIL01

RESPONSE=#@\$1

IXL150I 00.17.55 DISPLAY CF

COUPLING FACILITY SIMDEV.IBM.EN.0000000CFCC1

PARTITION: 0 CPCID: 00

CONTROL UNIT ID: 03C0

NAMED FACIL01

...

SENDER	PATH	PHYSICAL	LOGICAL	CHANNEL TYPE
80	A	ONLINE	ONLINE	CFS
CO	A	ONLINE	ONLINE	CFS

COUPLING FACILITY	DEVICE	SUBCHANNEL	STATUS
	4030	B 000E	OPERATIONAL/IN USE
	4031	B 000F	OPERATIONAL/IN USE
	4032	B 0010	OPERATIONAL/IN USE
	4033	B 0011	OPERATIONAL/IN USE

6.7 Failing and recovering a system with z/VM

The best way to simulate a system hang scenario is to issue the QUIESCE command on the system that you want to stop. Immediately, the message shown in Example 6-11 is displayed on the console of the crashed system.

Example 6-11 Messages after a QUIESCE

```
HCPPCW6533A Following is a priority message received by the service processor
- use the VINPUT command to respond:
BLW002I SYSTEM WAIT STATE 'CCC'X - QUIESCE FUNCTION PERFORMED
```

If SFM is *not* enabled, you should type the SYSTEM RESET command on the console of the failed system when the message is displayed.

This command is required to reset any outstanding I/O requests, CF requests, and reserves that might be outstanding from that system. Remember that if you are not using SFM, then you *must* reply DOWN to the IXC402D message, as shown in Example 6-12. The remaining systems in the sysplex do not start the cleanup for the failed system until this reply is provided.

Example 6-12 IXC402D message: Reply Down after system reset

```
16.02.04 #0$2 *812 IXC402D #0$1 LAST OPERATIVE AT 16:00:33.
REPLY DOWN AFTER SYSTEM RESET, OR INTERVAL=SSSSS TO SET A REPROMPT TIME.
...
```

If SFM is enabled, it issues the system reset automatically and we do not have to issue this command. Messages similar to those shown in Example 6-13 are displayed indicating that SFM has recognized that the system (named here as #0\$1) has stopped, and then showing that SFM has automatically partitioned that system out of the sysplex.

Example 6-13 SFM partitioning messages: Status update missing

```
IXC101I SYSPLEX PARTITIONING IN PROGRESS FOR #0$1 REQUESTED BY XCFAS.
REASON: SFM STARTED DUE TO STATUS UPDATE MISSING
...
IXC105I SYSPLEX PARTITIONING HAS COMPLETED FOR #0$1 391
- PRIMARY REASON: SYSTEM REMOVED BY SYSPLEX FAILURE MANAGEMENT BECAUSE
ITS STATUS UPDATE WAS MISSING
- REASON FLAGS: 000100
```

When you are ready to recover the failed system, follow the procedure shown in Example 6-14 on the console of the failed system.

Example 6-14 Recovering the failed system

IPL CMS

Hit “Enter” when the system enters VM READ mode

IPL xx where xx represents the load address of the z/OS guest

This IPLs CMS and then IPLs the z/OS operating system again. For a sample IPL EXEC, refer to Appendix B, “Sample REXX script to IPL z/OS systems on z/VM” on page 153.

6.8 References

Refer to the following for more information about the topics discussed in this chapter:

- ▶ *System/390 MVS Parallel Sysplex Continuous Availability Presentation Guide*, SG24-4502
- ▶ *z/VM V5R2.0 Running Guest Operating Systems*, SC24-6115-01
- ▶ *z/VM V5R2.0 CP Planning and Administration*, SC24-6083-03
- ▶ *z/VM V5R2.0 CP Commands and Utilities Reference*, SC24-6081-03
- ▶ *IBM Parallel Sysplex Trainer Environment*

z/TPF

In this chapter, we describe the benefits of running the System z Transaction Processing Facility, Enterprise Edition (z/TPF) as a guest under z/VM with the possibility to compile the z/TPF applications with Linux under z/VM.

In this chapter, we discuss the following topics:

- ▶ History and general overview of TPF
- ▶ Running z/TPF as a guest under z/VM
- ▶ z/TPF application compilation with Linux under z/VM

7.1 History and general overview of z/TPF

z/Transaction Processing Facility, Enterprise Edition V1.1 (z/TPF) is an IBM high-end System z based operating system designed specifically for the demanding transaction processing industry. z/TPF is based upon TPF 4.1, a platform that today is processing billions of transactions a year throughout the travel and finance industries. z/TPF, along with its corequisite database facility (z/TPFDF) brings forward an extremely robust solution to the very high-volume transaction processing arena, providing users with competitive marketplace advantages. z/TPF sets the standard for transaction processing capabilities.

7.1.1 History of TPF

z/TPF has a long and respected heritage. The story of z/TPF's origins reportedly began with a chance meeting in the 1950s between an executive in the travel industry, and an executive from IBM. This meeting, which is rumored to have occurred on an airplane, started discussions that led to the exploitation of computers in the business of airlines reservations. The problem that had to be addressed was the management of the inventory of available seats on available flights. Prior to this chance meeting, there had been some use of the emerging technology of computerization into the airline travel system, however, it was quite rudimentary. This meeting led to the development of the first computerized reservation systems.

During the late 1950s and early 1960s, IBM partnered with each of several airlines (American Airlines, Delta Airlines, and PanAm) to develop airline-specific solutions. These were the SABRE, the Deltamatic, and the Panamac systems. The release of the IBM S/360™ series of computers and the concept of a standard reusable platform was the catalyst that took the lessons and skills learned during these individual efforts and resulted in the Programmed Airline Reservation System (PARS); perhaps the first general purpose solution for managing reservations in the airline industry. PARS, and its worldwide version International Programmed Airline System (IPARS) contain the function of each the operating system component, and the applications layer.

It was not until 1969 that IBM split this merged system into two components, and focused on the operating system called the Airline Control Program (ACP). ACP was enhanced during the 1970s, primarily to take advantage of new hardware and networking technologies. Renamed to TPF in 1980, the product continued to see innovative solutions created to meet the availability and scalability demands of the ever expanding transaction-based solution.

This trend continued in the 1990s, with enhancements to the TPF platform being made to meet the need for connectivity, availability, scalability, and reliability.

Renamed to z/TPF as a result of the implementation of IBM z/Architecture, this product continues to innovate and evolve to help industries that need high end transaction computing solutions.

7.1.2 Overview of TPF

The origins of the TPF system can be traced back to systems created in the late 1950s to satisfy the requirements of airline reservation agents who accessed an inventory of available space on available flights for the purpose of selling tickets. A sale requires the deletion of a seat from the inventory of space and the creation of a passenger name record that accounts for the use of the seat.

This function, which is still a requirement today, is performed in real time and frequently occurs over two types of communication lines. A person representing an airline communicates with a customer by means of an ordinary telephone call, but accesses the necessary data using a terminal or workstation that is linked to a processing center through wide area communication facilities. The customer requesting a reservation is remote from the user (airline agent) who in turn is remote from the data.

Users within the vernacular of the TPF system are commonly called agents, and the data usage requests they generate are called (input) messages. The important TPF characteristic persists: The ability to accept unpredictable and very high message volumes at a processing center that gives business agents access to a shared centralized repository of information (database) that is updated in real time.

Any data processing environment that requires remote users to access shared information is a potential user of the TPF system. There are applications that can take advantage of the TPF system where the user and customer are identical, for example, a person utilizing an automatic teller machine. Shared data, in some instances, might simply be the information necessary to route messages to appropriate locations within a network of processing centers. Some examples of non-airline applications utilizing the TPF system are:

- ▶ Hotel reservations
- ▶ Credit authorization/verification
- ▶ Police car dispatching
- ▶ Electronic funds transfer switching
- ▶ Online teller memo posting
- ▶ Message switching
- ▶ Loan payment processing
- ▶ Communication transaction routers

The TPF system provides an extremely responsive solution to very high volume online processing that is required in many business enterprises. The TPF system

has wide acceptance within the airline industry, but is also used in non-airline applications for processing relatively simple inquiry and response messages associated with a large population of terminals and workstations. The TPF system is most commonly used for the purpose of accessing a large centralized database that is an inventory of business information.

The TPF system is a high-availability operating system designed to provide quick response times to high volumes of messages from large networks of terminals and workstations. A typical TPF system handles several hundred messages per second. A typical network varies from several hundred terminals and workstations to tens of thousands. The response time of the TPF system within a network is typically less than 3 seconds from the time the user sends a message to the time the user receives a response to that message. High availability is enhanced by the ability to quickly restart the system; restarting after a system failure takes between 30 seconds and 2 minutes.

These factors result in the unique design of the TPF system that can be summarized as:

- ▶ Efficient use of resources, such as main storage and file storage
- ▶ Short path lengths for critical system services, such as direct access storage device (DASD) input/output (I/O)
- ▶ Open-ended capacity growth, such as coupling as many as 32 multiprocessor Enterprise Systems Architecture (ESA) configurations with only a minimal increase in system overhead and expandable database capacity by the addition of DASD
- ▶ High throughput while maintaining quick response times
- ▶ High availability allows for 24-hour, 7-day a week operation
- ▶ Database integrity and online database maintenance capability

The TPF system is characterized by:

- ▶ Real time inquiry/response from geographically dispersed users
- ▶ Relatively short messages (in terms of length) in both directions
- ▶ Unpredictable message arrival rate
- ▶ Shared business data
- ▶ Database update during real-time operation
- ▶ Database maintenance during real-time operation
- ▶ Database integrity
- ▶ Duplicate data records for performance and reliability

- ▶ A communications interface for supporting a very large population of various types of terminals and workstations
- ▶ Fast response time per message
- ▶ High availability (24-hour real-time operation)
- ▶ System restarts that require only 2 minutes or less
- ▶ Dynamic monitoring of system operation (system measurement)

Note: For more information about z/TPF, see the following Web site:

<http://publib.boulder.ibm.com/infocenter/tpfhelp/current/index.jsp>

The IBM Redbook *z/TPF and WebSphere Application Server in a Service Oriented Architecture*, SG24-7309, is also a good source to get more information.

7.1.3 TPF application compilation with z/OS

TPF and z/TPF are unique from most systems in that they are target systems only. You do not develop on z/TPF but rather you do all your development and build on another platform and move the result to z/TPF for production.

In TPF and its predecessors, that build environment was MVS (later OS/390, now z/OS). That meant TPF and z/OS shared a compiler, language libraries, and a programming model. It also meant that development tooling was confined to what could work with z/OS and be customized for the target system approach.

7.2 Running z/TPF as guest under z/VM

This section provides some of the benefits of running z/TPF as a guest under z/VM. It also provides a description of the unique development and build environment used by z/TPF. We also describe the main tools used to build z/TPF programs.

7.2.1 Benefits of running z/TPF as guest under z/VM

When running the z/TPF as a guest under z/VM, you benefit from all the features that z/VM offers for running guest machines. This will make sense if there is already a System z hardware but so far not used for the z/TPF. As an add-on, you can use a Linux guest under the same z/VM image to run the compilation of the z/TPF applications.

7.2.2 z/TPF application compilation with Linux under z/VM

This section describes the open development environment of z/TPF and the main tools that are used to build z/TPF programs.

z/TPF open development

z/TPF is significantly different from the prior version of TPF in that the build environment is Linux, not z/OS. This can be any Linux. Except for building the z/TPF operating system itself, the development and build can be on any Linux server on any platform (Intel, Power or System z, for example), as long as the application does not use any assembly code. If the application utilizes assembly routines, development can only be done on Linux on System z, as it is the only platform where the IBM high-level assembler (HLASM) is supported.

z/TPF depends on the GNU tool chain, which includes the GNU C compiler (GCC) and common language libraries such as the GNU C library (glibc). Because GCC is a true cross-compiler, it can itself run anywhere and build for the target z/TPF system.

HLASM has experimental support for creating output that matches the Executable and Linking Format (ELF) used by Linux and z/TPF, but it is currently recommended that developers generate IBM Generalized Object File Format (GOFF) output files and then use the **goff2elf** utility to convert them into the proper format.

The term for shared libraries in Linux is shared objects; hence, all applications (assembler, C or C++) get loaded as shared objects. This, as well as the presence of POSIX API structures in z/TPF, means that the runtime environment as well as the build environment for the application programmer is extremely similar to Linux.

As a result, most packages built for Linux as well as programming skills are portable to z/TPF. The importance and impact of this cannot be understated. Saying “Linux like” is being conservative, as z/TPF and Linux shared the same compiler, libraries, tooling and linkage model. What is different is the memory model and database model, which supports the ability to do high-volume online transaction processing (OLTP).

Even the unique z/TPF database model need only be used for data with the highest requirements. Hundreds of thousands of lines of other open source code, are running on z/TPF with very few modifications, including Apache and OpenSSL.

The following commands are the main tools used to build z/TPF programs:

► **maketpf**

maketpf is the command used to assemble, compile, and link TPF programs, including online applications, CIMR components, keypoints, and offline utilities. **maketpf** is based on a schema that utilizes the GNU implementation of **make**, customized to support the TPF source code directory structure and provide a predefined set of assemble, compile, and link rules needed to build TPF systems code and applications.

► **bldtpf**

The **bldtpf** command is used to generate the z/TPF system configuration data (via SIP and FCTB), build the z/TPF online, offline, and utility programs, generate STUB and IPAT entries, and perform pat-to-control file conversions. These actions are driven using either the control files or the SIP decks as the primary input.

► **loadtpf**

loadtpf is the command used to generate an old or tld format loadfile to tape or hfs using the **offldr** program. If written to hfs, the resulting loadfile is optionally transferred to a user specified IP address via FTP for load to a z/TPF system. A load deck or load file can either be specified as input or can be generated from a control file, program name, or shared object name provided as input. This command is supported on Linux only. Also, the user name and password needed to log in to the specified IP address must be specified in a **.netrc** file.

7.3 Some more information about the TPF-gcc cross compiler

GCC stands for GNU Compiler Collection. It has been produced and maintained by the Free Software Foundation (FSF) since 1987.

GNU, Linux, and z/TPF

z/TPF is an operating system whose precursor is TPF 4.1. z/TPF and its predecessors have always been cross-developed systems, which means that the preparation of executables such as editing, assembly, compilation, linking, and all other offline steps, has never taken place on TPF itself. Instead executable preparation always takes place on another system, also called the build host, which has historically been either z/OS (or its predecessors MVS or OS/390) or z/VM.

The tools used to perform steps on the build host are typically called cross-assemblers or cross-compilers as applicable. The application of `gcc` that we are about to build for z/TPF development tasks is a cross-compiler.

We call a tool a cross when the system on which it runs differs from the system for which it is designed to distribute the code. For example, if we had a compiler that ran on an Intel machine but the code runs on a System z, this will be a cross-compiler.

There are cross-assemblers, cross-compilers, and cross-linkers. All of these are components in the so-called *toolchain*. The GNUPro toolchain is produced by Red Hat. Due to the flexible nature of the General Public License (GNU), Red Hat slightly modifies the components in the GNUPro toolchain to suit its development needs and those of its customers. At the moment, the GNUPro toolchain is only supported on those build hosts running on Linux platforms.

What is in the toolchain

It should be noted that a lot of development tooling comes with the z/TPF system. A large part of it consists of specially adapted and common tools from the GNUPro toolchain. However, some parts of the z/TPF GNU/Linux cross-development build host tooling are not part of the GNUPro toolchain. For example, `goff2elf` and `offldr` are IBM tools that have meaning only to z/TPF and are thus not part of the GNUPro toolchain. The GNUPro toolchain itself consists of every executable and data file needed to successfully run and test every part of itself. This inventory includes, but is not limited to:

- ▶ The linkage editor: `ld`
- ▶ The assembler: `as`
- ▶ The compiler driver: `gcc`
- ▶ The C compiler/front end: `cc1`
- ▶ The C++ compiler/front end: `cc1plus`

All of these items are compiled and linked from source code as a part of the build procedure.

7.4 Obtaining and using TPF-gcc

The GNUPro toolchain is available in two flavors: Prebuilt executables that are suitable for use on System z Linux or x86 Linux, or in source code form. z/TPF customers should contact their IBM Customer Service Representative (CSR) for details about obtaining either offering.

Both offerings are delivered in a compressed tape archive (`tar`), commonly referred to as a *tarball* or *tar file*. The source code for the toolchain as delivered

is portable and is usable for the build of many target systems. z/TPF is just one of the supported targets. You have to obtain one of these tarballs in order to build applications for z/TPF.

To compile and link z/TPF applications a sysroot is also required. The sysroot can be thought of as development libraries and headers for z/TPF applications. The sysroot can also be obtained from your CSR. GNUPro toolchain customers will have a tar archive file with the name `tpf-sysroot.tar.bz2` or similar.

The steps for installing GNUPro are beyond the scope of this book. You should consult the documentation that was provided with the offering for the latest procedures.

7.5 Post-installation steps

After you have GNUPro installed, you can verify the version that was installed with this command:

```
tpf-gcc -v
```

The command provides an output of the version of **tpf-gcc**, how it was configured, and other information about it. If the command does not produce an output with this information, you should check the installation documentation to verify that you have not made a mistake somewhere.

The following command has the gcc driver call the linkage editor with a command to produce an output of all the defaults under which it runs:

```
tpf-gcc -Wl,-v
```

7.5.1 Verifying the installation

Now that the TPF cross-compiler is installed, you can verify that it is functioning properly by creating a small test program and building it.

Note: The example that follows assumes some familiarity with UNIX development tools and also prior experience with the traditional z/TPF development process.

1. Create a working directory using the **mkdir** command as follows:

```
mkdir -p helloworld/base/rt
```
2. Create a file named `hello_tpf.c` in the `helloworld/base/rt` directory using your preferred text editor and enter the C source code shown in Example 7-1.

Example 7-1 The source for hello_tpf.c

```
#include <stdlib.h>
#include <stdio.h>
void HTPF () {
    printf ("Hello z/TPF\n");
}
```

3. Save and close the file. You now have to create a makefile that will tell the **maketpf** command what steps to execute to compile and link the application.
4. Create another file, this time named **htpf.mak** and located in the same **helloworld/base/rt** directory. The contents of the file are shown in Example 7-2.

Example 7-2 The htpf.mak makefile

```
APP := HTPF
APP_ENTRY := HTPF
APP_EXPORT := ENTRY
C_SRC := hello_tpf.c
maketpf.env := base_rt
include maketpf.rules
```

5. Save this file and create the final one, **maketpf.cfg**, in the **helloworld** directory. The contents of the file are shown in Example 7-3.

Example 7-3 The maketpf.cfg configuration file

```
TPF_ROOT := /root/helloworld
TPF_ROOT += /opt/tpfgcc/sysroot-tpf
TPF_BSS_NAME := BSS
```

Example 7-3 assumes that you were the root user when you created the **helloworld** directory and that you installed the **z/TPF** libraries and headers into **/opt/tpfgcc/sysroot-tpf**. If this is not the case, you should adjust those values accordingly.

6. It is now time to compile and link the sample application. The following commands accomplish this:

```
cd helloworld
maketpf /root/helloworld/base/rt/htpf.mak
```

The command should produce a shared object file named **HTPF.so** that can be loaded and run on a TPF system. The steps for doing so are beyond the scope of this book, but when run on a TPF system, the command should show the output “Hello z/TPF”.

The Communication Controller for Linux on System z

This chapter offers a brief introduction to the Communication Controller for Linux (CCL) on IBM System z and provides pointers to many of the CCL resources that are available.

8.1 What is the Communication Controller for Linux?

CCL on System z provides a platform to run the Network Control Program (NCP) software product in Linux on System z processors. This capability enables a migration path to move NCP-based Systems Network Architecture (SNA) workloads such as SNA Network Interconnect (SNI) from the IBM 3745 Communication Controller to System z servers.

CCL allows one to consolidate function from a 37xx Communication Controller onto one or more Linux images running under z/VM or in logical partition (LPAR) mode. An advantage to using CCL under z/VM is that multiple CCL images can easily be created and run simultaneously using shared hardware resources.

In addition to the costs saved by migrating NCP-based workloads from unsupported 37xx hardware, CCL can be run on Integrated Facility for Linux (IFL) engines as well as standard CPs, lowering the associated hardware costs even more.

8.2 Useful CCL resources

The primary resource for CCL information is the official IBM site for the product offering, which can be found at the following URL:

<http://www-306.ibm.com/software/network/ccl>

Other useful resources include:

- ▶ *IBM Communication Controller for Linux on System z V1.2.1 Implementation Guide*, SG24-7223
- ▶ *IBM Communication Controller Migration Guide*, SG24-6298
- ▶ *A Structured Approach to Modernizing the SNA Environment*, SG24-7334
- ▶ *IBM Communication Controller for Linux on zSeries V1R1 Implementation and User's Guide*, SC31-6872

z/VM system management tools

This chapter provides an overview of various system management tools that can be used to automate manual tasks of configuring, provisioning, and maintaining servers, operating systems, middleware software, and even custom-built applications.

In this chapter, we discuss the following topics:

- ▶ What is a system management tool?
- ▶ IBM Director
- ▶ IBM Tivoli Provisioning Manager
- ▶ Aduva OnStage
- ▶ Performance Monitor for z/VM
- ▶ Hobbit Monitor

9.1 System management tools

System management refers to the enterprise-wide administration of distributed computer systems. System management tools provide an enterprise approach in automating day-to-day operations at the same time improving the system utilization and support decisions with little to no manual intervention.

With a growing number of complex and heterogeneous IT environments in the industry, there is always a need for robust, flexible, and easy-to-use system management tools. Some of the functions of typical system management tools are:

► **Configuration management**

- Deploy and customize the hardware, software, middleware and user applications.
- Customize resources (storage, network, and processor setup).
- Creation and maintenance of user attributes and accounts in a consolidated administration mode.

► **Security management**

- Maintain and audit user credentials.
- Manage and maintain access to devices and services.
- Manage policies and rules applicable to software, middleware, and applications.

► **Fault management**

- Systematically diagnose the problems.
- Record events and other vital information about the operating system and applications running on the system.
- Data recovery during accidental data loss.

► **Performance management**

- Manage the performance and availability of the hardware and software.
- Record resource utilization for performance tuning.
- Automated resource allocations during peak loads.

9.2 IBM system management tools

Because the scope of system management covers a lot of areas in IT infrastructure administration, IBM provides a number of tools to easily manage the IT infrastructure of an organization. IBM offers both software and hardware tools for dealing with the various system management functions we just discussed.

System z servers running z/VM and Linux on System z provide important autonomic capabilities and rich functionality to help customers with the tools to make their technology responsive during outages.

System z products are designed to offer layer upon layer of fault tolerance and error-checking features. If a failure occurs, the built-in redundancy on the System z platform is intended to shift the work over from failing components to ones that work to prevent the user service from being interrupted. The failed components can be removed and replaced while the processor is still active so that service may continue.

For example, in System z, the Intelligent Resource Director (IRD) can adjust logical resources across logical partitions (LPARs) dynamically and automatically based on Workload Manager (WLM) and the policy defined for those LPARs. Because Linux runs on System z, it can exploit the system management features offered by System z hardware.

9.2.1 IBM Virtualization Engine

Virtualization Engine™ is an IBM offering that provides a single console to manage and integrate IT resources and services. It provides integrated services and technologies in the following areas:

- ▶ **Virtual resources:** To virtualize all the resources in the IBM server family (IBM System x™, IBM System i™, IBM System p™, and IBM System z)
- ▶ **Virtual management:** Tools include the IBM Enterprise Workload Manager (EWLM), Resource Dependency Services, and IBM Director
- ▶ **Virtual access:** Provides consistent interface access to virtualized resources

The IBM Virtualization Engine also provides an interface to resource health, resource topology, process management, resource dependencies, as well as launchpads for the IBM Director Console, the EWLM Control Center, and other platform-specific consoles.

9.2.2 IBM Director

IBM Director is delivered as a component of the IBM Virtualization Engine and Infrastructure services. The core of the Virtualization Engine contains the IBM Director suite of system management tools, which includes functionality for managing Linux on System z running within an LPAR or as a z/VM guest.

IBM Director can be described as an integrated, easy-to-use suite of tools that provides flexible systems management capabilities to help realize maximum system availability and help lower IT costs. With IBM Director, IT administrators can view and track the hardware configuration of remote systems in detail and monitor the usage and performance of critical components, such as processors, disks, and memory. The Director software components are:

- ▶ IBM Director server
- ▶ IBM Director agent
- ▶ IBM Director console
- ▶ Manageability access point (MAP)

IBM Director server

The server component contains the management data, the server engine, and the application logic. The server component and Director agent are installed by default during the installation procedure. The Director server provides basic functions such as:

- ▶ Discovery of managed systems
- ▶ Storage of configuration and management data
- ▶ Software and hardware inventory
- ▶ Event listening
- ▶ Security and authentication
- ▶ Management console support
- ▶ Administrative tasks

IBM Director agent

The Director agent allows the server to communicate with systems on which it is installed. The agent provides the server with management data, which can be transferred using the TCP/IP, NetBIOS, and IPX™ protocols.

IBM Director console

The Director console enables systems administrators to manage all systems that have an agent installed. This is done easily via the graphical user interface (GUI) by either a drop-and-drag action or a single click. Unlike the agent, the console and server communicate and transfer data using TCP/IP. The Director console does not require the Director agent to be installed unless you want to manage this system as well. The console does not require a license and can be installed on as many systems as needed.

Manageability access point

The z/VM manageability access point (MAP) implements a common information model (CIM) interface through which Director extensions can manage virtual servers and Linux guest systems. CIM is a standard of the Distributed Management Task Force (DMTF) for managing distributed resources. For more details about CIM, refer to the DMTF specification, which can be found online at:

<http://www.dmtf.org/standards/cim/>

9.3 z/VM Center

The z/VM Center is an extension to IBM Director that saves time and simplifies the creation and grouping of virtual servers, as well as the deployment of Linux systems under z/VM. With an extensive drag-and-drop interface, z/VM Center allows for fast, easy, and repeatable deployments of new virtual servers.

You can use the z/VM Center subtasks to virtualize System z hardware resources into z/VM virtual servers (guest virtual machines) and to deploy Linux instances on them.

The user has to know very little about z/VM to maintain virtual servers using z/VM Center; however, configuring z/VM Center and installing the Linux instances that serve as the source for the guest operating systems that you deploy requires working directly with z/VM.

The basic IBM Director functions (monitoring, event action plans, software distribution, task scheduling, and so on) can be exploited for virtual Linux servers under z/VM, but the z/VM Center extension provides two important tasks, which we are about to discuss in more depth.

Figure 9-1 demonstrates how the Director components are related.

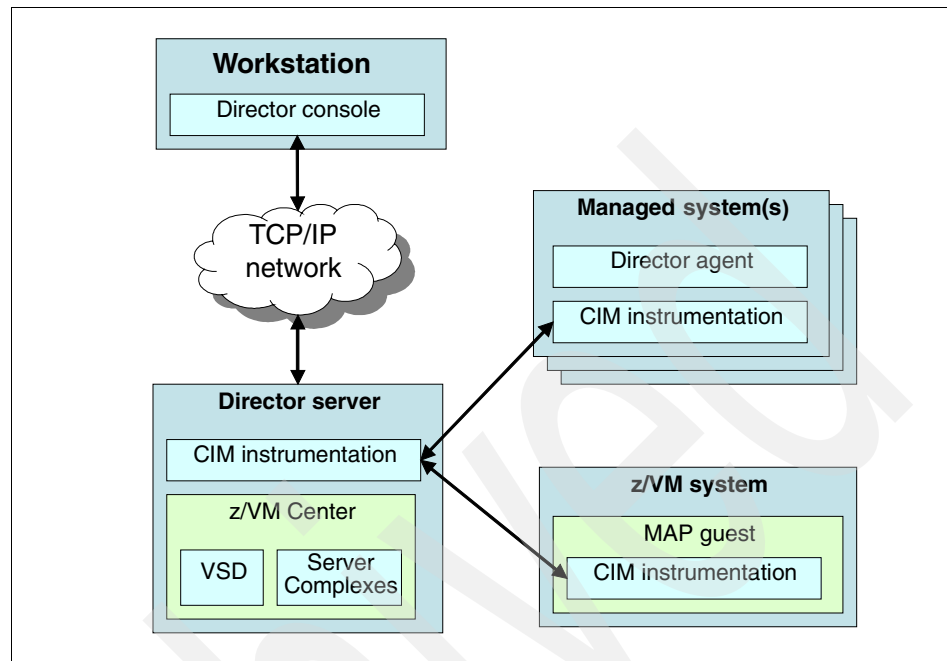


Figure 9-1 IBM Director Architecture

9.3.1 Server complexes

The server complexes task enables the consistent management of groups of Linux systems running within a z/VM environment. Groups of Linux systems defined by this task are referred to as a *server complex*.

Some of the advantages of using server complexes are:

- ▶ With the server complexes task, we can manage configurations of Linux guest systems. A Linux guest system is a combination of a Linux instance and the z/VM virtual server on which the Linux instance is installed. A server complex is a configuration profile for Linux guest systems and includes both Linux and z/VM aspects. A server complex can define network settings, Linux configuration scripts, disk access, and VM Resource Manager (VMRM) performance goals.
- ▶ We can automatically configure a Linux guest system by assigning it to a server complex. A new Linux guest system can also be created within a server complex. When creating a new Linux instance in a server complex, we automatically create a z/VM virtual server with a Linux instance that is configured according to the server complex. For creating a Linux guest

system, we require a virtual server template and an operating system template that have been created by the virtual server deployment task.

- ▶ We can make changes to a server complex and then apply the configuration changes to all Linux instances in the server complex.
- ▶ We can use server complexes to manage numerous Linux instances with similar configurations.
- ▶ Goal specification (such as CPU and direct access storage device input/output (DASD I/O)) using z/VM (VMRM) adjusts virtual machine CPU and I/O performance controls based on actual performance to attempt to achieve the goals associated with each workload.

9.4 Benefits of IBM Director and z/VM Center

The benefits of IBM Director and z/VM Center are:

- ▶ Unifies a heterogeneous IT infrastructure management through consistent and open standards-based with Linux on IBM System z.
- ▶ Safeguards data through centralized management while leveraging world-class System z security and reliability features.
- ▶ Simplifies the management of virtual Linux servers on the System z platform.
- ▶ IBM Director delivers a GUI that allows administration of z/VM without any specific z/VM knowledge.
- ▶ Automated, proactive capabilities that help reduce IT costs and maximize system availability.
- ▶ Provides a streamlined, intuitive user interface to get started faster and accomplish more in a shorter period of time.
- ▶ Open standards-based design and broad platform and operating support enable customers to manage heterogeneous environments from a central point.
- ▶ Inventory collection now collects firmware information about devices such as hard disk drives, Small Computer System Interface (SCSI) tape drives, network interface card (NIC) adapters, and RSA video BIOS for most managed systems.
- ▶ On the fly creation, configuring, and decommissioning of z/VM-based Linux guest systems.

9.5 IBM Tivoli Provisioning Manager

Tivoli Provisioning Manager (TPM) helps to provision, configure, and maintain the overall IT infrastructure (servers and virtual servers, operating systems, middleware, applications, storage and network devices) of an organization. Provisioning focuses on the self-configuring, dynamic allocation of individual elements of the IT infrastructure so that identities, storage, or servers are provisioned as business needs dictate. These elements can be:

- ▶ A single software package
- ▶ A software stack, which consists of a group of software packages
- ▶ A server that conforms to a template, which is a defined set of software and hardware resources

TPM is built on a service-oriented architecture (SOA), which enhances usability for executing changes while keeping server and desktop software compliant.

9.5.1 Tivoli Provisioning Manager components

TPM consists of the following components:

- ▶ Provisioning server
- ▶ Web-based operator and administrator console
- ▶ Automation Package Developer Environment

Provisioning server

The provisioning server is the server on which TPM is installed. The provisioning server contains the following components:

- ▶ **Provisioning database:** Physical database to hold the data model.
- ▶ **Data model:** A representation of all of the physical and logical assets that Tivoli Provisioning Manager manages. It keeps track of the data center hardware and associated allocations to applications, as well as changes to configuration.
- ▶ **Automation:** An automation package is a collection of workflow, scripts, and other commands and tools that apply to the operation of a specific type of software component or a physical device.

Prebuilt packages include automation packages to automate the provisioning of software, patches, images, and operating systems, and devices, including servers, network devices, and storage.

- ▶ **Compliance and remediation:** Allows us to examine the software and security setup that we have on a target computer in our infrastructure. If the

system is noncompliant, the recommendations on how to fix the issues are generated.

- ▶ **Reporting:** Reports allow you to retrieve current information about data center inventory, activity, and system compliance. TPM reporting functionality includes:
 - Numerous predefined reports
 - A Web-based query builder, which allows you to easily customize existing reports or create new reports
 - Easier access to information in the data model through more than 40 high-performance views
 - Easier sharing of report definitions through enhanced import and export capabilities in the Web interface
 - Charts and graphs
 - The ability to schedule reports to run at a later time including repeating intervals
 - E-mail report distribution and notification
 - Integration with third-party reporting software
- ▶ **Discovery:** This component provides automated processes that allow you to find resources, as well as any changes to existing resources
- ▶ **Deployment Infrastructure:** This component supports reconfiguring and reallocation of resources in our managed environment.

Web services interface

Web services, including Web Services Resource Framework (WSRF) services, allow you to access the TPM data center model directly rather than launching the Web interface. Using the Web services you can access, manipulate, or change objects directly in the data center model.

Operator and administrator console

The Web-based operator and administrator console allows you to interact with the Tivoli Provisioning Manager provisioning server. The operator and administrator console provides a graphical representation of the data center, includes wizards to simplify configuration, and other features such as reporting and task status tracking that are not available from the command-line interface.

Automation Package Developer Environment

The Automation Package Developer Environment (APDE) is an Eclipse-based plug-in environment that the automation package developers can use to customize existing automation packages or create new automation packages.

9.5.2 Benefits of IBM Tivoli Provisioning Manager

The benefits of the Tivoli Provisioning Manager are:

- ▶ With the prebuilt automation packages, we can quickly deliver value with automation of common tasks for leading products.
- ▶ The development toolkit enables building of automation packages tailored to the environment and IT or business processes.
- ▶ With the integration of Tivoli Intelligent Orchestrator, Tivoli Provisioning Manager can enhance utilization and IT service delivery.
- ▶ Tivoli Provisioning Manager enables easily adapts to growing and changing IT environments.
- ▶ Customized workflows can well implement any companies IT practices and procedures.
- ▶ With automation of tasks, IT administrators can manage more resources with fewer people, allowing key skilled people to focus on activities that grow the business rather than serve to maintain it.
- ▶ On a testing environment, Tivoli Provisioning Manager manages and maintains a reliable test environment that help testers to identify the source of code defects more quickly and retest with confidence.

Tip: For more information about the Tivoli Provisioning Manager, visit the following Web site:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v13r1/index.jsp>

9.6 Aduva OnStage

Aduva OnStage is an integrated tool that deploys, provisions, and manages Linux-based systems and applications. It supports multiple Linux distributions running on mainframes to distributed environments.

For more information about Aduva OnStage, see the following Web site:

http://www.aduva.com/index.php?page=onstage_overview

9.7 Performance toolkit

The Performance toolkit for VM is an enhanced real-time performance monitor, which allows systems programmers to monitor system performance and to analyze bottlenecks. The toolkit can help systems programmers to make more efficient use of system resources, increase system productivity, and improve user satisfaction.

In addition to analyzing VM performance data, the performance toolkit for VM processes Linux performance data obtained from the IBM Resource Management Facility (RMF™) Linux performance gatherer, `rmfpm`s. The Linux performance data obtained from RMF can be viewed and printed similarly to the way VM data is presented.

9.7.1 Performance toolkit functions

The functions provided by the performance toolkit for VM includes:

- ▶ Operation of the system operator console in full-screen mode
- ▶ Management of multiple z/VM systems (local or remote)
- ▶ Post-processing of performance toolkit for VM history files and VM monitor data captured by the MONWRITE utility
- ▶ Viewing of performance monitor data using either Web browsers or PC-based 3270 emulator graphics
- ▶ TCP/IP performance reporting

With performance toolkit, the following resources can be monitored:

- ▶ General CPU performance
- ▶ System and user storage utilization and management
- ▶ Channel and I/O device performance, including cache and SEEKs analysis data
- ▶ Detailed I/O device performance, including information about the I/O load caused by specific minidisks on a real disk pack
- ▶ General user data: resource consumption, paging information, IUCV and VMCF communications, wait states, response times
- ▶ Detailed user performance, including status and load of virtual devices
- ▶ Summary and detailed information about Shared File System servers
- ▶ Configuration and performance information for TCP/IP servers
- ▶ Linux performance data

9.7.2 Modes of operations

The following are the different modes in which a performance toolkit can be operated:

- ▶ **Basic command mode**, usually referred to as basic mode, is the normal operating mode that allows you to enter CP and CMS commands. This mode immediately displays any messages received from CP.
- ▶ **Re-display mode**: The console log created while operating in basic mode can be browsed in re-display mode and specific strings (for example, USERIDs) can be searched using a LOCATE facility.
- ▶ **Performance monitoring mode** displays data on CPU, I/O, and user performance collected by this machine. It allows real-time performance analysis of z/VM systems.
- ▶ **Remote performance monitoring mode** displays performance data collected by the performance monitors from another virtual machine on the same or other VM systems.
- ▶ **Internet interface** allows retrieval and displays the same performance data via a standard Web browser.
- ▶ **Monitor data scan mode** displays performance data from a MONWRITE disk file, created by z/VM systems.
- ▶ **Trend file scan mode** displays performance history data from a trend file.

9.7.3 Linux monitoring with performance toolkit

To monitor Linux internal performance, the following components are required:

- ▶ The performance toolkit for VM must be installed, configured, and active.
- ▶ DDS interface must be installed and active. More information about the DDS can be found at:

<http://www.ibm.com/servers/eserver/zseries/zos/rmf/rmfhtmls/pmweb/pm1in.html>

- ▶ Performance data must be collected, stored, and managed on the Linux system.

Important: For further information about the performance toolkit, see the IBM Redbook *Linux on IBM eServer zSeries and S/390: Performance Toolkit for VM*, SG24-6059.

9.8 Hobbit monitor

Hobbit is a tool for monitoring servers, applications, and networks. It provides a simple, intuitive way of checking the health of your systems from a Web browser, and can also alert you to any problems that arise. It collects information about the health of your computers, the applications running on them, and the network connectivity between them.

Hobbit collects information through querying network services (Web, Lightweight Directory Access Protocol (LDAP), DNS, Mail, and so on) or from scripts that run either on the Hobbit server or on the systems that you monitor. The Hobbit package includes a Hobbit client, which you can install on the servers that you monitor. Then all of this information is presented in a set of simple, intuitive Web pages that are updated frequently to reflect changes in the status of your systems. Hobbit is capable of monitoring various network services, local server application logs, resource utilization, and much more.

At the same time, Hobbit stores trend and availability information about everything it monitors. Therefore, we can always analyze the system's behavior for over a period time. Through Hobbit's extensive notification framework, the users can define criteria for when and at what situation it has to send out an alert.

For monitoring Web applications, it is possible to not only check that the Web server is responding, but also that the response looks correct by matching the response against a predefined pattern or a checksum.

9.8.1 Features of Hobbit monitor

Some of the features of Hobbit monitor are:

- ▶ Hobbit provides a flexible and extensible monitoring system through which we can monitor thousands of hosts and networks with ease.
- ▶ Hobbit can be configured to notify the monitoring data through e-mail and SMS.
- ▶ Because it is open source, Hobbit is extensible through external scripts and clients.
- ▶ Networked applications can be monitored, and connections to a particular port can be checked for response.
- ▶ Monitoring and testing of common network protocols can be performed.
- ▶ Hobbit is capable of monitoring various network services, local server application logs, resource utilization, and much more.

9.8.2 z/VM client extension to Hobbit monitor

The Hobbit client for z/VM and VM/ESA runs on virtual machines, and performs tests on a VM system and reports the status of those tests back to the Hobbit server. The client is written in REXX and uses TCP/IP sockets support to send the status messages to the Hobbit server.

For installing and configuring z/VM client on a virtual machine, a VM user must be created, with the CLASS D privilege. The privilege is required for the z/VM client to execute various commands for gathering information about the resources. Hobbit z/VM client can be configured via files on the virtual machines. Using HOBVMTST configuration file, tests can be included or eliminated, just by commenting and uncommenting the options. The HOBVARS file basically lists Hobbit z/VM client's configuration details, such as Hobbit server address, port, CPU test threshold values, and so on.

Hobbit z/VM client extension can perform the following tests on the virtual machine:

- ▶ CPU utilization information is gathered using the CP IND VM command.
- ▶ Uses CSL routines to monitor SFS file pools and optionally filespace within those pools.
- ▶ Uses the CP IND command to extract the current paging rate.
- ▶ Compares a preconfigured list of machine names to the output of the QUERY NAMES command and reports if any virtual machines in the list are not running (this mimicks the functionality of the procs test on Linux systems).
- ▶ Uses the CP Q ALLOC PAGE command to determine the page space utilization.
- ▶ Uses the CP Q ALLOC SPOOL command to determine the spool space utilization.

Attention: For further information about the Hobbit monitor and z/VM client, visit the following links:

- ▶ Official Web site:
<http://sourceforge.net/projects/hobbitmon/>
- ▶ Documentation on Hobbit available at:
<http://hobbitmon.sourceforge.net/>
- ▶ z/VM client can be downloaded at:
http://www.vmassist.com/rs_samples/

Porting applications to Linux on System z

In this chapter we discuss about application porting considerations and a porting tool that can be used for porting applications for Linux on zSeries.

In this chapter, we discuss the following topics:

- ▶ Application portability
- ▶ Porting consideration for Linux on zSeries
- ▶ Migration Toolkit for Linux (Solaris™ to Linux)

10.1 Application portability

Porting is basically adapting an application program from the operating environment in which it was developed to another operating environment so that it can be used in the new environment. Porting applications from one environment to another normally requires sizable effort and specialized skills.

Because Linux application programming interfaces (APIs) are identical to that of other UNIX flavors and if the application does not require any kernel information, architecture-specific informations or extensions, those applications can be directly treated as source compatible.

In this case, the migration just requires the application to be moved on the Linux on zSeries environment and recompiled. Generally, Java-based Web applications can be easily ported, because they are extremely portable across all the platforms and operating systems.

Another technique is to use GNU-based C and C++ compilers that are common across UNIX flavors and are widely supported. Using various cross-compilation techniques, the applications can be compiled using these compilers on the source platform for the target Linux platforms. This makes the porting easier because all the language-specific issues can be solved leaving only the changes specific to APIs that have to be dealt with.

10.1.1 Linux on System z porting considerations

There are several considerations that have to be addressed during an application port for Linux on System z.

- ▶ The application that contains any architecture or assembler-specific code has to be re-implemented for S/390 architecture. Opcodes have to be changed to S/390 opcodes or if the code uses assembler header files, we require a S/390 version of the header.
- ▶ All of the middleware software, libraries, and databases used by the application have to be ported/built specifically for S/390 Linux. Linux does not support sharing of synchronization objects such as mutexes, semaphores, and conditional variables between different processes (not threads).
- ▶ zSeries is a big endian system. Any code that processes byte-oriented data that originated on a little endian system might require some of its bytes to be swapped. The data might have to be regenerated or, if that is not possible (for example, shared files), the application might have to be reworked to adjust for processing little endian data.

Note: Big endian and little endian refer to which bytes are most significant in multi-byte data type.

- ▶ On a migration involving Web applications, most of the contents can be directly moved without any change to any Web servers running on Linux for zSeries.
- ▶ Web applications that are Windows-based and if they constitute some custom-built DLLs, then the applications have to be recoded using compatible scripting language (Perl) and then have to be ported on Linux for System z.
- ▶ For applications that use Oracle® and DB2 as their back-end, then the databases can be directly moved to Linux on System z. In case of other databases (SQL server), there is a need to build a similar or more advanced Linux-compatible database. And then we can move the data from the SQL server to the newly built Linux-based database.

Tip: For more information about various other porting considerations that you have to take, visit the following Web site:

http://www-128.ibm.com/developerworks/eserver/articles/linux_s390/index.html

10.2 Migration Toolkit for Linux on System z

At present for porting applications developed on other platforms to Linux, we are required to go through documentations describing the various porting considerations that have to be taken into account. These documents give an overview, but do not make recommendations that are specific to the application that a customer is considering to port on Linux.

Knowing the difficulties in porting an application, IBM offers a toolkit for Linux that provides a single pass, API checker that is available through open source, with the intended customers being those who plan to do their own port of an application.

The tool scans through the source code (C, C++ files), looks for constructs/language support, APIs that might have to be modified during the port, and makes recommendations as to how to modify the code. The tool does not automatically make changes to the source code, but rather suggests alternatives in areas that are likely to cause problems during a porting effort.

10.2.1 Components of Migration Toolkit

The Migration Toolkit for Linux has the following major components:

- ▶ GUI Front-end
- ▶ Code Scanner

10.2.2 GUI Front-end

The graphical user interface (GUI) allows the user to invoke a code scan function and view the output. The scan function scans the code source files of given directory tree and generates copies of those files for C/C++ applications. The user can then view the output files through the GUI provided. The flagged functions are highlighted and color coded when viewed through the tool.

Important: The tools do not change the application code, but identifies where code changes are required to run your Solaris code, shell scripts, and makefiles on Linux.

With the metrics function, percentages of flagged functions, by level of difficulty number (Low, Medium, High or Unknown), and percentage of lines flagged can be obtained. This gives the programmer or developer an idea about the difficulty of the port. The knowledge base provides description of the API, alternative solutions (if any), coding examples, and recommendations on how to modify the code while porting.

The tool also provides additional information about build and compiler issues when porting from Solaris to Linux. For making changes to the code, clients can use the summary view as a guide. The copies of source code that this tool creates have an extension of portout. These portout files contain a copy of the original code with markers around recognized porting issues.

The metrics view of the tool helps the customers to assess the amount of effort involved in the porting project. The metrics include the total number of lines in the project files scanned, the number of issues identified, and a breakdown of those issues by level of difficulty.

10.2.3 Code Scanner

This component scans each C/C++ source input file in the specified tree and looks for lines to flag. The file extensions that can be scanned by the tool are: .c, .h, .l, .y, .C, .H, .L, .cc, .cpp, .cxx, .hxx, .hpp. Assembly language files (.s and .S files) are also processed, except that scanning for the function name or include

files are not performed. It uses various input files to determine whether a function, pragma statement, or #include line file is to be flagged.

Important: For more information about Migration Toolkit, its components, usage, and other details, visit the following IBM Web site. This toolkit is available as a free download (requires an IBM PartnerWorld® ID):

http://www-1.ibm.com/partnerworld/pwhome.nsf/weblook/pat_linux_migrate_solaris.html

10.2.4 Benefits of Migration Toolkit for Linux

The benefits of the Migration Toolkit for Linux are:

- ▶ Migration Toolkit optimizes, reduces, and automates the tedious task of porting.
- ▶ The metrics view of the migration tools can be used to understand and identify where to focus efforts to enhance the code in the new Linux environment.
- ▶ With the help of this tools, we can quickly migrate or assess the technical issues that will be faced in moving applications from other operating systems to Linux on IBM eServer platforms.
- ▶ Migration Toolkit provides GUI-based view, where the functions are flagged and colored according to their level of difficulty. With these things, developers can accelerate the porting process, resulting in greatly reduced time, effort, and risk of a port.
- ▶ With in-depth suggestions, examples, and references provided by the tool, developers can always use them as a reference in their porting effort.

Appendixes

The appendixes contain sample scripts that you can use to perform some of the procedures listed in this book, as well as information about obtaining electronic copies of the scripts.

Sample scripts to use when cloning systems

This appendix contains several scripts that you can use to update the configuration files for a cloned Linux system and also to copy the direct access storage device (DASD) volumes using FlashCopy.

For directions on how to download these scripts, see Appendix C, “Additional material” on page 157.

Copying production volumes

Use the REXX script in Example A-1 to attach all nine of the production volumes to the current user, link the nine test volumes, and then use FlashCopy to copy the production volumes to the test volumes.

Example: A-1 COPYPROD EXEC

```
/* COPYPROD EXEC */

/* Attach the WPS production volumes. */
SAY "Attaching WPS volumes"
ATTACH F00-F02 *

/* Link the first volume, flashcopy it, and then detach it. */
SAY "Copying WPS volumes"
LINK LINWPS01 F00 A00 MR
FLASHCOPY F00 0 END TO A00 0 END
DETACH A00

/* Repeat with the 2nd... */
LINK LINWPS01 F01 A01 MR
FLASHCOPY F01 0 END TO A01 0 END
DETACH A01

/* and 3rd volumes. */
LINK LINWPS01 F02 A02 MR
FLASHCOPY F02 0 END TO A02 0 END
DETACH A02

/* Detach the WPS production volumes. */
SAY "Copying complete, detaching WPS volumes"
DETACH F00-F02

/* Attach the DB2 production volumes. */
SAY "Attaching DB2 volumes"
ATTACH F03-F05 *

/* Link the first volume, flashcopy it, and then detach it. */
SAY "Copying DB2 volumes"
LINK LINDB01 F03 A03 MR
FLASHCOPY F03 0 END TO A03 0 END
DETACH A03

/* Repeat with the 2nd... */
```

```
LINK LINDB01 F04 A04 MR
FLASHCOPY F04 0 END TO A04 0 END
DETACH A04
```

```
/* and 3rd volumes. */
LINK LINDB01 F05 A05 MR
FLASHCOPY F05 0 END TO A05 0 END
DETACH A05
```

```
/* Detach the DB2 production volumes. */
SAY "Copying complete, detaching volumes"
DETACH F03-F05
```

```
/* Attach the load balancer production volumes. */
SAY "Attaching load balancer volumes"
ATTACH F06-F08
```

```
/* And do the same thing we did with the WPS and DB2 volumes. */
SAY "Copying load balancer volumes"
LINK LINSPRAY F06 A06 MR
FLASHCOPY F06 0 END TO A06 0 END
DET A06
```

```
LINK LINSPRAY F07 A07 MR
FLASHCOPY F07 0 END TO A07 0 END
DETACH A07
```

```
LINK LINSPRAY F08 A08 MR
FLASHCOPY F08 0 END TO A08 0 END
DETACH A08
```

```
/* Detach the load balancer production volumes. */
SAY "Copying complete, detaching volumes"
DETACH F06-F08
```

```
SAY "Done copying"
```

Updating Linux network settings

Use the script in Example A-2 to update the IP address and host name for a Red Hat Enterprise Linux (RHEL) or SUSE Linux Enterprise Server (SLES) Linux system.

Example: A-2 fix-network.sh

```
#!/bin/bash

function fix_hosts()
{
    sed -i.old -e "{ s/${oldhostname}/${newhostname}/g ; s/${oldip}/${newip}/g }" \
    /etc/hosts
    echo "Changed /etc/hosts, backing up as /etc/hosts.old"
}

function setup_rhel()
{
    cd /etc/sysconfig/network-scripts

    if [ -z "$(ls ifcfg-eth*)" ]; then
        echo "No /etc/sysconfig/network-scripts/ifcfg-eth* files found, exiting."
        exit 2
    fi

    for ifcfg in "$(ls ifcfg-eth*)"
    do
        sed -i.old "s/${oldip}/${newip}/" $ifcfg
        mv ${ifcfg}.old old-${ifcfg}
        echo "Changed $ifcfg in /etc/sysconfig/network-scripts,"
        echo "    backing up as old-${ifcfg}"
    done

    cd /etc/sysconfig
    if [ -e "network" ]; then
        sed -i.old "s/${oldhostname}/${newhostname}/" network
        echo "Changed /etc/sysconfig/network, backing up as network.old"
    fi

    fix_hosts
}
```

```

function setup_sles()
{
    cd /etc/sysconfig/network

    if [ -z "$(ls ifcfg-qeth-bus-ccw-*)" ]; then
        echo "No /etc/sysconfig/network/ifcfg-qeth-bus-ccw-* files found,
exiting."
        exit 2
    fi

    for ifcfg in "$(ls ifcfg-qeth-bus-ccw-*)"
    do
        sed -i.old "s/$oldip/$newip/" $ifcfg
        mv ${ifcfg}.old old-${ifcfg}
        echo "Changed /etc/sysconfig/network/${ifcfg}, backing up as
old-${ifcfg}"
    done

    cd /etc
    if [ -e "HOSTNAME" ]; then
        sed -i.orig "s/$oldhostname/$newhostname/" HOSTNAME
        echo "Changed /etc/HOSTNAME, backing up as /etc/HOSTNAME.old"
    fi

    fix_hosts
}

if [ "$#" -ne "2" ]; then
    echo "usage: $0 <new hostname> <new ipaddress>"
    exit 1
fi

newhostname=`echo $1 | cut -d'.' -f 1`
newip=$2
oldhostname=`hostname | cut -d'.' -f 1`
oldip=`ifconfig eth0 | grep 'inet addr:' | cut -d':' -f 2 | cut -d' '
-f 1`

```

```

if [ -e "/etc/redhat-release" ]; then
    setup_rhel
elif [ -e "/etc/SuSE-release" ]; then
    setup_sles
else
    echo "Unsupported or unrecognized distribution."
    exit 3
fi

```

Updating WebSphere Portal Server instance data

Use the script in Example A-3 to update the instance data for WebSphere Application Server and WebSphere Portal Server with the new host name. It was originally included with *WebSphere Portal Server for Linux on zSeries and z9*, REDP-4175, under the name searchreplace.

Example: A-3 update-wps.sh

```

#!/bin/bash
function help
# give help
#+=====+
{
echo "Usage: update-wps.sh oldhostname newhostname"
exit
}
#+=====+
main()
if [ $# != 2 ]; then
echo "Error: incorrect number of arguments"
help
fi
OLDHOST=$1
NEWHOST=$2
#+=====+
# find all the files from here on that contains the old host name
that's not a
fgrep -R $OLDHOST * | grep -v Binary > outlist
#+=====+
# now replace those files that contain the old host name with the new
host name
echo "Replacing files that contain $OLDHOST with $NEWHOST"
cat outlist | cut -d ":" -f 1 > outlist2

```

```

cat outlist2 | while read myline; do if [ -f "$myline" ]; then mv
"$myline" "$myline.bak"; sed "s/$OLDHOST/$NEWHOST/g" "$myline.bak" >
"$myline"; rm "$myline.bak"; fi; done
#+-----+
touch newdirs
# now find those directories that are the old host name
find ./ -name "*$OLDHOST*" > olddirs
while [ "`cat olddirs | wc -l`" -gt 0 ]
do
cat olddirs | sed "s/$OLDHOST/$NEWHOST/g" > newdirs
exec 3<> olddirs
exec 4<> newdirs
# Change directories with old host name to new host name
read myline <&3; read myline2 <&4; echo "Moving $myline to $myline2";
mv "$myline" "$myline2"
exec 4>&-
exec 3>&-
find ./ -name "*$OLDHOST*" > olddirs
done
#+-----+
# clean up
rm outlist
rm outlist2
rm olddirs
rm newdirs

```

Sample REXX script to IPL z/OS systems on z/VM

This appendix contains a REXX script that you use to IPL the z/OS systems in a Parallel Sysplex running on z/VM. In fact, you can use this EXEC to IPL any z/OS guest running on z/VM.

IPL EXEC

Example B-1 shows a sample IPL EXEC, which can be used to IPL the z/OS guest systems running on z/VM. The EXEC can be downloaded from the IBM Redbooks Web site and copied to the first virtual machine running the z/OS system in the sysplex. Change the name to IPL1 EXEC. When “IPL1” is typed at the “CMS” command prompt, an IPL of the first system in the Parallel Sysplex will be performed as will each guest.

Example: B-1 Sample IPL EXEC script for IPL of z/OS guests in a Parallel Sysplex running on z/VM

```
/* ipl exec */
trace 'o'
'vmfclear'
cp = 'EXECIO * CP (SKIP STRING'
cp_stem = 'EXECIO * CP (STEM CP_LINE. STRING '
/*****/
/*   DEFINE SOME VALID TSO SCREEN ADDRESSES   */
/*****/
cp 'DEF GRAF 8E2 3270'
cp 'DEF GRAF 8E3 3270'
cp 'DEF GRAF 8E4 3270'
cp 'DEF GRAF 8E5 3270'
cp 'DEF GRAF 8E6 3270'
cp 'DEF GRAF 8E7 3270'
cp 'DEF GRAF 8E8 3270'
cp 'DEF GRAF 8E9 3270'
cp 'DEF GRAF 8EA 3270'
cp 'DEF GRAF 8EB 3270'
cp 'DEF GRAF 8ED 3270'
cp 'DEF GRAF 8EE 3270'
cp 'DEF GRAF 8EF 3270'
/*-----*/
/* Virtual CTCA's */
/*-----*/
CP 'COUPLE 4A11 TCPIP 4A11'
CP 'COUPLE 4A12 TCPIP 4A12'
cp_stem 'Q V CONS'
parse value cp_line.1 with . cons_addr . 'VSM' cons_type
cmd = 'CP SYSTEM RESET' || '15'X
if cons_addr ^= '08E0' then
do
  cmd = cmd || 'DET' cons_addr || '15'X
  cmd = cmd || 'DEF cons 8E0 3270' || '15'X
```

```

end
cmd = cmd||'TERM CONMODE 3270'||'15'X
cmd = cmd||'SET RUN ON'||'15'X
cmd = cmd||'SET MACH ESA'||'15'X
cmd = cmd||'DEF STOR 768M'||'15'X
say '*****'
say '*'
say '* This is a z/OS V1R4 system,'
say '* WAIT, the #@$1 system is coming up....'
say '*'
say '*****'
cmd = cmd||'IPL A843 LOADPARM 1D00FKM1 CLEAR'
/*cmd = cmd||'IPL A843 LOADPARM C730V1M1 CLEAR' */
push cmd

```

It might be a good idea to keep the z/VM virtual machine name to be same as the z/OS guest name. This helps to define a single EXEC, which can be named as PROFILE EXEC, for automatic IPL. Such a PROFILE EXEC can be put on a “master” 191 disk (the CMS A disk) to be linked as the 191 disk for all the guests that intend to use this PROFILE EXEC. This way the PROFILE EXEC can be maintained from a single place (for example, from the 191 disk of the “master” user “ZOSMAINT”, which is discussed in 6.2.3, “Establishing data sharing among the z/OS virtual machines in a virtual Parallel Sysplex” on page 100). Also, managing a single EXEC is much more easier than managing different EXECs for different guests.

Additional material

This IBM Redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this IBM Redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG247355>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG24-7355.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<i>File name</i>	<i>Description</i>
SG247355.zip	Zipped Code Samples

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	2 MB minimum
Operating System:	Linux or Windows XP

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder. Upload the REXX executables to your z/VM machine and run from a CMS command line.

Related publications

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

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 162. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *System/390 MVS Parallel Sysplex Continuous Availability Presentation Guide*, SG24-4502
- ▶ *IBM Communication Controller Migration Guide*, SG24-6298
- ▶ *The IBM TotalStorage DS8000 Series: Concepts and Architecture*, SG24-6452
- ▶ *Linux for IBM System z9 and IBM zSeries*, SG24-6694
- ▶ *IBM Communication Controller for Linux on System z V1.2.1 Implementation Guide*, SG24-7223
- ▶ *z/TPF and WebSphere Application Server in a Service Oriented Architecture*, SG24-7309
- ▶ *A Structured Approach to Modernizing the SNA Environment*, SG24-7334
- ▶ *WebSphere Portal Server for Linux on zSeries and z9*, REDP-4175

Other publications

These publications are also relevant as further information sources:

- ▶ *z/VM V5R2.0 General Information*, GC24-6095-04
- ▶ *MVS System Commands*, SA22-7627
- ▶ *zSeries Platform Test Report for z/OS and Linux Virtual Servers*, SA22-7997-02
- ▶ *z/VM V5R2.0 CMS Commands and Utilities Reference*, SC24-6073-01
- ▶ *z/VM V5R2.0 CP Commands and Utilities Reference*, SC24-6081-03
- ▶ *z/VM V5R2.0 CP Planning and Administration*, SC24-6083-03

- ▶ *z/VM V5R2.0 Running Guest Operating Systems*, SC24-6115-01
- ▶ *z/VM V5R2.0 System Operation*, SC24-6121-01
- ▶ *XEDIT User's Guide*, SC24-6132
- ▶ *IBM Communication Controller for Linux on zSeries V1R1 Implementation and User's Guide*, SC31-6872
- ▶ *z/VSE V3R1.1 Planning*, SC33-8221-02
- ▶ *z/VSE V3R1.0 Operation*, SC33-8239-01
- ▶ *Linux on zSeries - Device Drivers and Installation Commands*, SC33-8282
- ▶ *Linux on System z - Device Drivers, Features, and Commands*, SC33-8289-02
- ▶ *Linux on System z - How to Improve Performance with PAV*, SC33-8292

Online resources

These Web sites are also relevant as further information sources:

- ▶ IBM z/VM Operating System
<http://www.vm.ibm.com/index.html>
- ▶ The z/VM Internet Library
<http://www.ibm.com/eserver/zseries/zvm/library/>
- ▶ Ten great reasons to run Linux as a guest of z/VM
<http://www.vm.ibm.com/linux/benefits.html>
- ▶ z/VM resources for Linux on IBM System z
<http://www.vm.ibm.com/linux/>
- ▶ *z/VM and Linux on IBM System z: The Virtualization Cookbook Version 2*
<http://www.linuxvm.org/present/index.html>
- ▶ IBM Directory Maintenance for z/VM
<http://www.vm.ibm.com/related/dirmaint/>
- ▶ IBM TPF Product Information Center
<http://publib.boulder.ibm.com/infocenter/tpfhelp/current/index.jsp>
- ▶ VM Parallel Access Volumes Support
<http://www.vm.ibm.com/storman/pav/pav2.html>

- ▶ WebSphere Portal Server Configuration properties reference
http://publib.boulder.ibm.com/infocenter/wpdoc/v510/index.jsp?topic=/com.ibm.wp.ent.doc/wpf/inst_prep.html
- ▶ Configuring WebSphere Portal for DB2
http://publib.boulder.ibm.com/infocenter/wpdoc/v510/topic/com.ibm.wp.ent.doc/wpf/cfg_db2.html
- ▶ Updating the node configuration file (UNIX)
<http://publib.boulder.ibm.com/infocenter/db2luw/v8/topic/com.ibm.db2.udb.doc/start/t0006350.htm>
- ▶ Linux for S/390 and zSeries porting hints and tips
http://www-128.ibm.com/developerworks/eserver/articles/linux_s390/index.html
- ▶ Migration Kit for Solaris OS to Linux
http://www-1.ibm.com/partnerworld/pwhome.nsf/weblook/pat_linux_migrate_solaris.html
- ▶ Communication Controller for Linux on System z
<http://www-306.ibm.com/software/network/cc1>
- ▶ Tivoli Provisioning Manager documentation
<http://publib.boulder.ibm.com/infocenter/tivihelp/v13r1/index.jsp>
- ▶ RMF PM with Support for Linux Enterprise Server
<http://www.ibm.com/servers/eserver/zseries/zos/rmf/rmfhtmls/pmweb/pm1in.html>
- ▶ DMTF: Common Information Model (CIM)
<http://www.dmtf.org/standards/cim/>
- ▶ Linux Channel Bonding
<http://sourceforge.net/projects/bonding>
- ▶ Quagga Routing Suite
<http://www.quagga.net/about.php>
- ▶ Aduva Web site
http://www.aduva.com/index.php?page=onstage_overview
- ▶ Hobbit monitor
<http://sourceforge.net/projects/hobbitmon/>

- ▶ Documentation on Hobbit
<http://hobbitmon.sourceforge.net/>
- ▶ z/VM client download site
http://www.vmassist.com/rs_samples/

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

A

Address Resolution Protocol (ARP) 28
Aduva OnStage 123, 132
Airline Control Program (ACP) 112
American Standard Code for Information Interchange (ASCII) 73, 77
application portability 138
Automation Package Developer Environment (AP-DE) 130–131

B

Benefits of Migration Toolkit for Linux 141
BLK 70
blue color 53, 56, 59, 66, 69

C

CF service machine 17–18, 99–100
 OPTION directory control statement 17
 user ID 17
channel bonding 21, 38–40, 42
 Benefits 42
Channel Report Word (CRW) 50
channel-to-channel adapter (CTCA) 14
code defect 132
Code Scanner 140
command output 52–53
Communication Controller for Linux 122
components of Migration Toolkit 140
connectivity failure (CF) 97, 99, 102–105, 107
Control Program (CP) 4, 12, 16, 19, 22, 24–25, 30–32, 50, 85, 100–101, 103, 105, 107–108, 110
 single instance 16
Conversational Monitor System (CMS) 4
COUPLING FACILITY
 SIMDEV.IBM.EN 102–107
 SPACE Utilization 103
Coupling Facility (CF) 15, 17–18, 50, 97, 99, 102–105, 107, 109
 Control Code 17, 103
 service machine 17
 virtual machine 16, 102, 104, 108
CTC adapter 9, 14, 67

D

DASD device 51–53, 55, 67, 70, 100
 entire list 54
 I/O activity 54
DASD Dump Restore (DDR) 84, 90
DB2 instance 82, 93
DETACH A00-A02 92
direct access storage device (DASD) 49, 51–53, 55, 66–67, 70–71, 83, 114
directory entry 30, 32, 51–52, 57, 60, 66, 68–69, 71, 88–90
dynamic activation 63
dynamic routing 21, 39, 42–48
 configuration 46
 example 44
 VIPA 42

E

Edge Components (EC) 82
Enterprise Systems Architecture (ESA) 114
Ethernet mode 28, 32
external security manager (ESM) 6, 76, 88

F

file system 54
Fixed Block Architecture (FBA) 70
FLASHCOPY command 90–91
Free Software Foundation (FSF) 117

G

general availability (GA) 7
GNUPro toolchain 118
 common tools 118
graphical user interface (GUI) 127
Group Control System (GCS) 4
guest LAN 11, 21–26, 29, 33
 benefits 26
 external connectivity 11
guest operating system 3, 12, 24–25, 49–51, 53–55, 57–58, 62, 66, 70, 73–74, 83, 100, 110
guest system 9, 13, 15, 19–20
 flexible communication 19

GUI front-end 140

H

Hardware Configuration

Manager 4

hardware configuration

definition 4

Hardware Management Console (HMC) 77

hash policy 39

HiperSockets network

concentrator 21, 33–34, 36

IP subnet 35

script 36

Hobbit monitor 135

Hypervisor 3, 12, 15, 83

I

I/O activity 54–55

I/O device 6, 19

IBM Director 123, 125–127, 129

benefits 129

IBM System

z9 129

IBM Tivoli 123, 130, 132

benefits 132

integrated ASCII 77

Integrated Facility for Linux (IFL) 11

Intelligent Resource Director (IRD) 125

IP address 23, 27, 29, 33–36, 45, 47, 60, 93

IPL CMS 67, 100–101

IXC518I System 104, 107

L

Lightweight Directory Access Protocol (LDAP) 135

Linux 4–5, 9–11, 13–15, 51, 54–55, 58–60, 62–65, 72, 81, 83–86, 92, 111, 116–118, 125, 127–128, 132

Linux distribution 38, 63, 83, 132

Linux guest 13–15, 22–24, 26–27, 31, 33, 37, 53–55, 58–60, 62–65, 77, 85–86, 95, 115

command output 54

CPU removal 65

CPUs present 63–64

DASD device offline 54

network interface 60

newly defined CPU 62

operating system 62

system 63–64, 128–129

transparent, high-speed data access 13

very high-performance networking 13

Linux image 31

Linux instance 127–128

Linux on System z

Migration Toolkit 139

porting considerations 138

Linux server 10, 37, 116

high availability solution 37

Linux system 30, 34, 49, 60, 63, 65, 77, 82, 127–128

boot/config-xxxx file 63, 65

logical partition (LPAR) 83

logical volume manager (LVM) 55

LPAR 10–11, 77, 82–83, 125–126

M

MAC address 27–28, 32, 40

system-wide range 32

MACHINE ESA

15 101

2 61, 66

4 68–69

Management Access Point (MAP) 126–127

manual SC24-6115-01 18, 20

Message Facility 15–16, 18

software simulation 18

MII Status 40

multi-processor (MP) 18

N

network adapter

failure VIPA 48

fault tolerance 37

network control program (NCP) 122

network interface card (NIC) 22–23, 25–26, 30, 32, 55–60

NIC detail 56, 59

NICDEF statement 25, 32

non-airline application 113–114

O

Operating System

other vital information 124

operating system 4–5, 8, 49–51, 53–55, 57–58, 62, 66, 70, 112, 114, 116–117, 124, 129

- new release 5
- OPERATIONAL/IN Use 103, 106, 108
- OSA device 23, 25, 31
- OSA Express 29–30, 33, 37
 - adapter port 30
- overcommit ratio 83

P

- Parallel Sysplex 12, 16, 97, 99–100, 102
 - environment 12
- PARTITION
 - 0 CPCID 102, 104–107
- password quagga 46
- performance toolkit 5, 76
- potential problem 90
- press PA1 105, 107
- production environment 3–4, 12, 25, 33, 42, 82
- production system 9, 14, 88, 90, 92, 104
 - duplicate name issues 104
- PROFILE Exec 61, 68
- Programmed Airline Reservation System (PARS) 112
- puts the console (PC) 103, 105, 107

R

- Redbooks Web site 162
 - Contact us xiii
- removed NIC
 - z/VM directory entry 57
 - z/VM directory statement 57
- Resource Access Control Facility (RACF) 5, 75, 88
- running Linux 13
- running TPF 15
- running z/OS 15
- running z/VM 20
- running z/VSE 19–20

S

- sample z/VM directory
 - entry 60, 66, 68, 100
 - profile 101
 - statement 71
- Secure Sockets Layer (SSL) 6
- server complex 128–129
 - Linux instances 129
 - new Linux guest system 128
- SNA Network Interconnect (SNI) 122

- System Delivery Offering (SDO) 4
- system management
 - API 76
 - tool 123
- system simulation 9
- SYSTEM VSWITCH1 56–57
- System z 4, 10–11, 21, 42, 84–85, 92, 121, 125–126
 - SUSE Linux Enterprise Server 92
- Systems Network Architecture (SNA) 122

T

- T=0.01/0.01 22
 - 50
 - 08 52
- TCP/IP stack 23, 42
- testers tester 132
- Tivoli Provisioning Manager (TPM) 132
- TPF system 113–114, 116, 118
 - potential user 113
 - response time 114
- Transparent Services Access Facility (TSAF) 4

U

- uniprocessor (UP) 14, 16–18

V

- VDISK creation 70
 - user limit 70
- VIPA address 42, 44
- virtual machine (VM) 4–5, 11, 14, 16–20, 22, 30–31, 33, 49–51, 53, 55, 58, 60–63, 66–67, 69, 71, 74, 100, 102, 107
 - directory entry 57
 - dynamic I/O changes 50
 - LAN segment 30
 - OPTION directory control statement 18
 - other groups 22
 - sample z/VM directory entry 60, 68
 - z/VM directory entry 69
- virtual network 22, 24–26, 33
- virtual processor 14, 61, 68
- Virtual Switch
 - failover 55
 - SNMP agent 75
 - support 11
 - traffic 75

- VSWITCH1 31
- Virtual Telecommunications Access Method (VTAM) 4
- VM virtualization 5, 10–11
- VM/VSE Interface 19
- VM/VSE interface 19
- VMLAN ACNT 25
- VMLAN statement 25
- VSWITCH 21, 27–32, 37, 40, 56, 58, 87–88, 90
- VSWITCH controller 30–31

W

- Web Services Resource Framework (WSRF) 131
- Web Site 157
- WebSphere Portal Server 82, 93–94, 161

Z

- z/OS 100–105, 107, 110
- z/OS image 5, 16, 100, 102
- z/OS system 15, 87, 101
- z/OS virtual machine
 - Sample z/VM directory entry 102
- z/OS virtual machine (z/VM) 100
- z/VM
 - guest systems 15
 - system management APIs 75
- z/VM 5.2 74–75
- z/VM 5.3 74–77
 - Control Program 74–75
 - general availability 74
 - Improvements 74
 - installation 74
- z/VM Center 127, 129
- z/VM feature 6
- z/VM guest
 - name 92
 - operating system 66
 - support 11, 16, 75, 77
 - system 58–59
- z/VM Linux guest 54, 58, 60, 63–64
 - additional CPUs 63
 - DASD device 54
 - DASD device online 54
- z/VM System
 - delivery offering 4
- z/VM Version 5
 - offers new level 10
 - pricing model 11
- Release 2 4
- Release 3.0 73–74
- Release 5.2 7
 - running 11
- z/VM virtual machine 51–54, 56–57, 60–62, 66, 68–69
 - command output 61, 69
 - current configuration 51
 - DASD device 53–54
 - defined CPUs 61
 - new CPU 62
 - zSeries server 125

Using z/VM for Test and Development Environments: A Roundup

(0.2" spine)
0.17" <-> 0.473"
90 <-> 249 pages



Redbooks

Using z/VM for Test and Development Environments: A Roundup

How guest systems benefit from z/VM virtualization technology

Exploiting z/VM functionality for your guest systems

Testing a Parallel Sysplex under z/VM

This IBM Redbook shows the strengths of z/VM and how you can use these strengths to create a highly flexible test and production environment.

Some of the strengths of z/VM that are shown in this book are that you can run Linux on z/VM, you can run a sysplex under z/VM, and you can develop code under z/VM for z/TPF. You can also provision Linux guests under z/VM. A vswitch allows you to connect all of your guests (all operating systems that run under z/VM) easily to the network.

You can simulate your production environment on a sysplex. The intention of this book is to show the strengths of z/VM and how you can use these strengths to simulate your production environment and expand your application development and testing environments.

**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