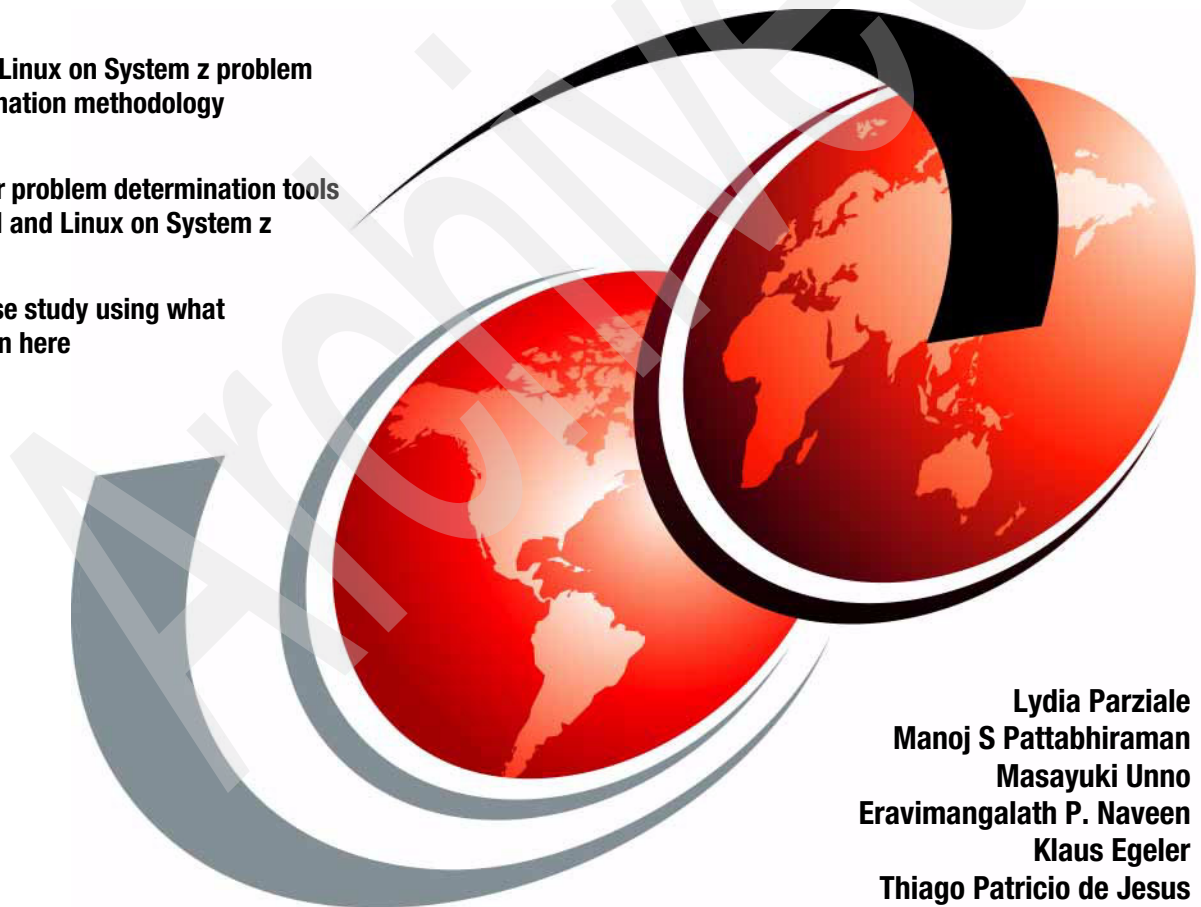


Problem Determination for Linux on System z

Learn a Linux on System z problem
determination methodology

Discover problem determination tools
for z/VM and Linux on System z

Do a case study using what
you learn here



Lydia Parziale
Manoj S Pattabhiraman
Masayuki Unno
Eravimangalath P. Naveen
Klaus Egeler
Thiago Patricio de Jesus



International Technical Support Organization

Problem Determination for Linux on System z

August 2008

Archived

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

First Edition (August 2008)

This edition applies to Version 5, Release 3, of z/VM (product number 5741-A05), Novel USE Linux Enterprise Server 10 and Red Hat Linux 5.

© Copyright International Business Machines Corporation 2008. 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
This book The team that wrote this book	xi
Become a published author	xiii
Comments welcome	xiii
Chapter 1. Problem determination methodology	1
1.1 Proactive approach for minimizing the effects of problems	3
1.1.1 Design and implementation phases	3
1.1.2 Test phase	7
1.2 Problem determination basics	9
1.2.1 Basic problem determination steps	9
1.2.2 Gather information and explore the type of problem and its trigger .	11
1.2.3 Analysis of a problem	13
1.2.4 Collect additional information	15
1.2.5 Escalate to the support team to analyze	16
1.2.6 How to approach complex problems	16
1.3 Problem resolution and follow-up actions	17
1.3.1 Expanded check for other systems	18
1.3.2 Update	18
1.4 Summary	20
Chapter 2. Problem determination tools for z/VM	21
2.1 z/VM system tools	22
2.1.1 Collecting technical setup information	22
2.1.2 Indicate	23
2.1.3 query srm	27
2.1.4 query alloc page	28
2.1.5 VM trace	28
2.1.6 vmdump	33
2.2 IBM z/VM Performance Toolkit	38
2.2.1 Modes of operation	39
2.2.2 Selecting performance screens	43

2.3 IBM Tivoli OMEGAMON XE on z/VM and Linux	52
Chapter 3. Problem determination tools for Linux on System z	59
3.1 Information-gathering tools	60
3.1.1 dbginfo.sh	60
3.1.2 vmstat	61
3.1.3 System status (sysstat) tool	62
3.1.4 top	66
3.1.5 ps	67
3.1.6 ipcs	67
3.1.7 iostat	68
3.1.8 netstat	68
3.1.9 DASD statistics	69
3.1.10 OProfile	73
3.1.11 zFCP statistics	76
3.1.12 Summary of information-gathering tools	80
3.2 Dump and trace	80
3.2.1 When to use dump	81
3.2.2 When a trace should be run	82
3.2.3 Stand-alone dump	83
3.2.4 Verification of the dump	84
3.2.5 Core dump	87
3.2.6 Summary of dump techniques	92
3.3 Debugging tools	92
3.3.1 gdb	93
3.3.2 strace	96
3.3.3 ltrace	98
3.3.4 SysRq	100
3.3.5 s390dbf	104
3.3.6 zFCP/SCSI	106
Chapter 4. Network problem determination	109
4.1 Overview of networking options for Linux on System z	110
4.1.1 Open Systems Adapters	110
4.1.2 HiperSockets	112
4.1.3 Guest LAN	113
4.1.4 Virtual Switch	113
4.1.5 Summary of networking options for Linux on System z	114
4.2 Network interface detection and configuration under Linux on System z	115
4.2.1 The Linux hotplug system	115
4.2.2 Flow of network interface detection and activation under Linux . .	116
4.2.3 Network interfaces and their interface names	119

4.2.4 Network configuration files overview	121
4.3 Network problem determination	123
4.3.1 Problems related to network communication	123
4.4 Case studies	159
Chapter 5. Performance problem determination	163
5.1 What a performance problem is	164
5.2 General flow of performance problem determination	165
5.2.1 Data necessary for investigation	169
5.2.2 CPU bottlenecks	170
5.2.3 Memory bottlenecks	183
5.2.4 I/O bottlenecks	192
5.2.5 Case studies	205
Chapter 6. Storage problems	211
6.1 How to determine a storage-related problem	212
6.1.1 How to check DASD devices	213
6.1.2 How to check FCP/SCSI devices	215
6.1.3 What to check for multipathing	218
6.1.4 What to check for LVM	219
6.1.5 I/O statistics.	223
6.2 Case study	223
Chapter 7. Eligibility lists	233
7.1 CP scheduler.	234
7.2 SRM controls.	237
7.2.1 SRM STORBUF	237
7.2.2 SRM LDUBUF.	238
7.2.3 QUICKDSP	239
7.3 Other recommendations	239
Chapter 8. Hardware-related problems	243
8.1 Basic configuration of hardware	244
8.2 Device recognition and management	244
8.2.1 Common I/O layer.	245
8.2.2 Kernel layer.	247
8.2.3 Device drivers	250
8.2.4 Confirming status of channel path and devices	251
8.3 Hardware problem determination	255
8.3.1 Initial determination of hardware or software problems	255
8.3.2 Investigating hardware-related messages	257
8.3.3 Approach for hardware problems	260
8.3.4 After determining whether problem is software or hardware.	263

8.4 Case study	263
8.4.1 Memory card	263
8.4.2 OSA	264
8.4.3 DASD errors	265
8.4.4 Network interface error upon reboot	266
8.4.5 DASD recognition problem in boot process	267
Chapter 9. Installation and setup problems	269
9.1 Understanding the installation process	270
9.2 Gathering information regarding installation process	275
9.3 Scenarios	282
9.3.1 Trouble during the installation	282
9.3.2 Trouble after the installation	283
Chapter 10. Booting problems	285
10.1 Understanding the boot process	286
10.1.1 IPL phase	287
10.1.2 Boot loader phase	287
10.1.3 Kernel phase	290
10.1.4 Init phase	293
10.2 Where to gather information regarding boot process	297
10.3 Commands	305
10.4 Boot problem examples	306
10.5 Scenarios	308
10.5.1 IPL failure	308
10.5.2 Failing in kernel booting	309
10.5.3 Failing since the init phase	309
10.5.4 Failure to start a service	311
Chapter 11. Case study: slow responding Web site	313
11.1 Three-tier Web application environment	314
11.1.1 Trade 6 Web serving application	314
11.1.2 Our Web serving setup	315
11.1.3 WebSphere Studio Workload Simulator	317
11.2 Symptoms of the problem	317
11.2.1 Investigation of the problem	320
11.2.2 Post investigation tuning	322
11.2.3 What if the load is increased - does the problem reappear	322

11.3 z/VM to the rescue	323
11.4 Emergency scanning.....	329
Related publications	333
Online resources	333
How to get Redbooks.....	334
Help from IBM	334
Index	335

Notices

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

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

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

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

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

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

COPYRIGHT LICENSE:


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

Trademarks

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

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

AIX®
DB2®
developerWorks®
ECKD™
ESCON®
eServer™
FICON®
GDDM®
HiperSockets™

IBM®
Lotus®
OMEGAMON®
OS/390®
Redbooks®
Redbooks (logo) ®
REXX™
S/390®
System z™

System z9®
Tivoli®
TotalStorage®
WebSphere®
z/OS®
z/VM®
z9™
zSeries®

The following terms are trademarks of other companies:

SUSE, the Novell logo, and the N logo are registered trademarks of Novell, Inc. in the United States and other countries.

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

J2EE, Java, JavaServer, JVM, Voyager, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

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® Redbooks® publication addresses some of the common problems that some customers have experienced on the Linux® on System z™ platform. Our intention is to demonstrate the characteristics of common problems and the process one would follow to determine the root cause of the problem and thus resolve the problem.

We start with a discussion on how to approach problem solving in the Linux on System z environment.

Additionally, we list the tools that are available to assist in diagnostics and discuss when you would use which tool.

This book The team that wrote this book

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

Lydia Parziale is a Project Leader for the ITSO team in Poughkeepsie, New York, with domestic and international experience in technology management including software development, project leadership, and strategic planning. Her areas of expertise include e-business development and database management technologies. Lydia is a Certified IT Specialist with an MBA in Technology Management and has been employed by IBM for 24 years in various technology areas.

Manoj S Pattabhiraman is a Staff Software Engineer in Linux Technology Center (LTC), India. He holds a master's degree in computer applications from the University of Madras. He has seven years of experience in z/VM® Application Development and Systems Programming for Linux on zSeries®. Apart from his Linux on System z development activities, he also provides technical support for various Linux on System z customers. Manoj has contributed to several z/VM and Linux on System z related IBM Redbooks publications, and has been a presenter at ITSO workshops for Linux on System z. His areas of expertise include z/VM, Linux on System z, middleware performance, and z/VM REXX™ programming.

Masayuki Unno is a Technical Sales Specialist for System z in Japan and Asia-Pacific countries. His mission is technical support for Linux on System z (zLinux) and includes client-facing support. His areas of expertise include zLinux

high-availability system design (z/VM, Linux with network, storage), system management (both of z/VM and LPAR-native Linux), and troubleshooting.

Eravimangalath P. Naveen is an IT Infrastructure Architect at the IBM India Software Lab in Bangalore with eight years of experience. His areas of expertise include IBM AIX®, Linux, z/VM and System z hardware, IBM Tivoli® Storage Manager, VMware, Storage Systems, and Networking and Virtualization across all IBM Server Systems. He has been a presenter at ITSO workshop events related to Linux and IBM System z.

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. Klaus has contributed to several z/VM-related and Linux-related IBM Redbooks publications, and has been a presenter at ITSO workshops and customer events.

Thiago Patricio de Jesus is an Advisory IT Specialist at Lotus® Software in Brazil. He has seven years of experience in the software group field. He holds a degree in electrical engineer with a specialization in Electronics and a Technology and Information Systems *Latu Senu* specialization from Universidade Santa Cecilia. His areas of expertise include Linux and Collaboration Software.

Special thanks to the following people for their contributions to this project:

Roy P. Costa
International Technical Support Organization, Poughkeepsie Center

Val Nixon
ATS Tivoli Support, IBM USA

Steffen Thoss, Holger Smolinski, Susanne Wintenberger
IBM Böblingen

Lester Peckover
IBM UK

Noriyuki Samejima and Naoshi Kubo
System z ATS, IBM Japan

Chris Borntraeger
IBM Böblingen

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will 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 will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

- ▶ Use the online **Contact us** review Redbooks 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



Problem determination methodology

This chapter describes the approach for problem solving. Note that the approach to resolving problems is related not only to actions after problems happen. We also take a proactive approach and follow-up actions to prevent problems from occurring in the future.

Figure 1-1 shows the typical life cycle for problem determination, where you would either start when the problem happens or, when taking a proactive approach, design and manage your system with problem prevention in mind.

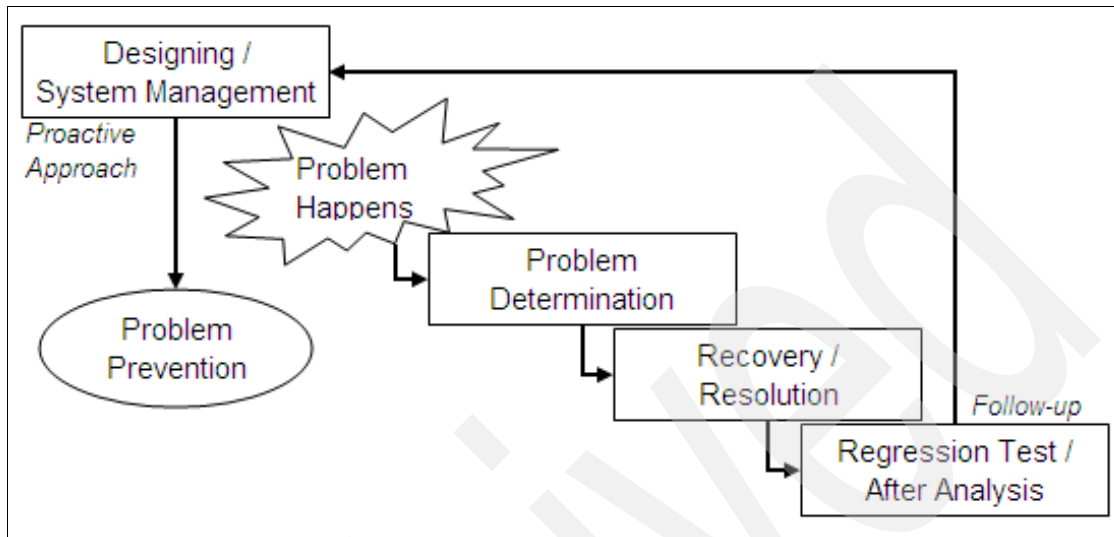


Figure 1-1 Life cycle of problem determination

1.1 Proactive approach for minimizing the effects of problems

Approaching problems proactively is an important methodology.

Sometimes, we cannot react to problems quickly because of a lack of preparation. This impacts the resolution of problems and may remain as risks on your customer's system. An example of problems that could be avoided and thus minimize impact to your customer's system are no package for `dbginfo.sh`, no performance data, no disk for stand-alone dump, and so on.

Designing, system management, and preparation for problems minimizes the affects of problems and shorten the total turn around time (TAT) by providing a quick response when a problem happens. Quick responses require preparation during the design and test phases.

1.1.1 Design and implementation phases

The following three points should be considered during the design and implementation phases:

- ▶ Preparation for recovery from a single point of failure (SPOF), if one exists
- ▶ Monitoring system design for a daily gathering of information
- ▶ Installation and settings for emergency information collection

Note that it is difficult to implement these in the later phase of test or after system integration, so planning ahead for troubleshooting is important.

Preparation for recovery from SPOF failure

Basically, redundant configurations should be adopted on each component as often as possible. These are very important for Linux on System z because of availability improvements. But if there are any single points of failure (SPOFs), prepare the quick recovery procedure for these problems. For example, if OSAs are not redundant, recovery procedures for network devices are needed.

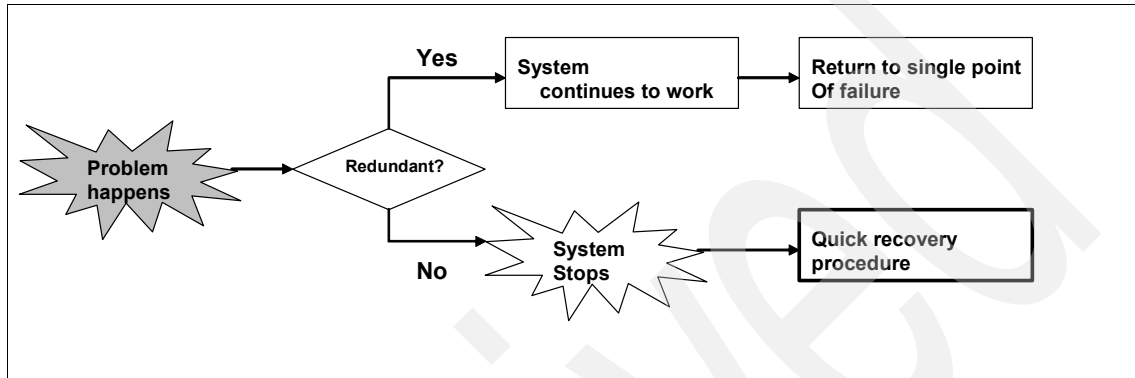


Figure 1-2 Necessity of quick recovery procedure

A rescue Linux image will help recover your system when a problem happens. For example, if a boot problem happens because of a configuration error, a rescue Linux image will let you check and change the configuration files that caused a *file system mounting failed* error.

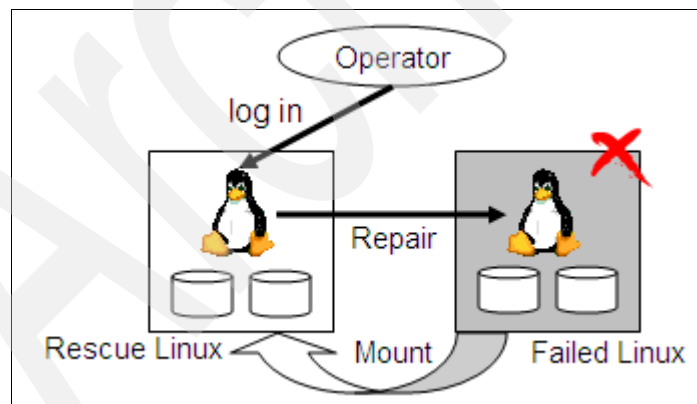


Figure 1-3 Sample image of recovery with rescue Linux

Monitor system design

Monitoring the system will also help you when trouble happens. For example, security traceability is important for finding mis-operations, not only for security

incidents. An operator can notice trouble by monitoring critical message alerts. Performance monitor data helps to find the cause of service-level degradation.

Table 1-1 is a sample template of system management monitoring requirements. Note that these are not all required for all customers. You can discuss these items based on your customer's Service Level Agreement (SLA).

Table 1-1 Monitor contents template

A) Data collection	A-1) Data contents	A-1-1) Security Traceability	Collecting logs that can trace the authority exceeded operation (who, when, where, what).
		A-1-2) Capacity Management	Collecting the performance data of H/W, OS, and processes for analyzing capacity trends for future planning.
		A-1-3) Problem Traceability	Collecting time-line logs of each problem and incident, which helps analysis of the problem.
	A-2) Automated keeping		Storing the corrected data automatically in specified terms. Removed the expired data automatically.
	A-3) Data processing		For performance data, it should be easy to process for analyzing, for example, organized on an Excel® sheet.
B) System monitoring	B-1) Monitoring contents	B-1-1) Performance monitoring	Monitoring the performance status based on threshold (for example, overconsumption system resource.
		B-1-2) Service monitoring	Monitoring the status of service and system. Alert the operator if these stop or the condition becomes bad.
		B-1-3) State transition monitoring	Alert the operator about a status transition (for example, takeover on redundant system, batch job ends successfully).
	B-2) User interface		The operator can monitor easily with graphs or charts, and so on.
	B-3) Service duration		Monitoring 24x365.

Tip: While we are discussing the monitoring of contents, we also discuss how to control command responses, including error messages (such as from Linux commands and shell scripts, z/VM commands, and REXX). A command response is the starting point for recognizing that something may be happening in the system. We recommend that command responses should not be discarded but instead be redirected to /dev/null.

Preparation for gathering information for troubleshooting

Here we discuss what kind of emergency information may be needed. We recommend preparing a script that gathers emergency information needed for any problems. The scripts listed in Table 1-2 are an example of scripts used to gather troubleshooting information. Your scripts should include dbginfo.sh, as well as saving the system log (syslog).

You may need to prepare ahead of time to gather information at this phase. Some of the preparation you should think about doing is:

- ▶ Install package selection for Linux tools.
- ▶ Have disk equipment ready in case you need to take a stand-alone dump.
- ▶ Be aware of the settings needed for data gathering (core dump and so on).
- ▶ If you are going to run scripts under z/VM, you should also enable the perfsvm user to be able to collect z/VM monitor raw data and follow the recommendation from the z/VM team at:

<http://www.vm.ibm.com/perf/tips/collect.html>

Table 1-2 Packages of popular problem determination tools

Commands	Packages
dbginfo.sh	s390utils on RHEL4.4 (or later), no need on SLES
gcore, gstack	gdb
strace	strace
tcpdump	libpcap
lsof	lsof
iostat, sar	sysstat
ps, top, vmstat	procs

Other sections of this book describe each component and tool listed in Table 1-2.

1.1.2 Test phase

You can rehearse for problems and practice gathering information in the test phase. These two actions will provide valuable information:

- ▶ Failure simulation tests
- ▶ Practicing gathering information

Failure simulation tests

Failure tests should be done during the test phase as often as possible. You can check the behavior, logs, and recovery procedures for each potential problem. If target components are redundant, you should check that they work correctly. These tests may include hardware, software, and application failure. Tests for procedures for physical component replacement are also good.

Check these items for a failure test:

- ▶ System behavior
 - The system behavior is the same as expected.
 - The extent of impact is the same as expected.
- ▶ Resource usage
 - The resource usage is the same as expected.
 - The resource usage increase has side effects:
 - Performance decline of own application
 - Performance decline of the other systems that share the same resources
 - Causes a critical issue (for example, CPU overload, I/O hang, thrashing, and so on)
- ▶ Redundancy check

The system continues or recovers as expected.
- ▶ Error code and messages
 - The error code and messages are the same as expected.
 - Choose messages that should be monitored.
 - The critical message is described in messages and code manuals.
- ▶ Recovery procedure

The recovery procedure works correctly.

Table 1-3 represents some of the potential problems that can be practiced with during the test phase. The question you should ask for each is what is the expected behavior if each of these happens. You would want to see what the logs would say, what error codes and messages are given, and if and how you would recover from each of these incidents.

Table 1-3 Test case sample

Network	OSA cable pull/plug	Server	Linux down
	OSA chpid offline		LPAR stop
	Physical switch failure		z/VM user down
	VSWITCH failure		z/VM down
Storage	Fibre cable pull/plug	Tape	Driver failure
	Channel chpid offline/online		Library failure
	Director/SAN switch failure	Software	Process failure
	RANK failure		Application failure

Practice collecting data

Practicing gathering troubleshooting data is very important. You can check that needed information is gathered correctly. Also, you need to be able to prepare a quick procedure to gather the necessary information. You may want to perform these rehearsals during the failure tests mentioned previously.

We suggest confirming the following during your rehearsals:

- ▶ Operation
Practice using the IBM z/VM Performance Toolkit (see 2.2, “IBM z/VM Performance Toolkit” on page 38) and the Linux sysstat package (see 3.1.3, “System status (sysstat) tool” on page 62). For example, you could create some z/VM raw monitor data and read it into the IBM z/VM Performance Toolkit.
- ▶ The data that is gathered is correct.
- ▶ Additional system load - Your system should have enough of each of the following in order to collect the data that you need:
 - CPU and memory
 - File size (enough for file systems)
 - Duration time
- ▶ Know your contact point and formations in an emergency.

- ▶ Organizing the system information
 - Version list of OS, Microcode, Middleware, Packages, PTF, and so on
 - H/W configuration diagram

Execute the rehearsal when the system is both active and in a failed state. This will help you to understand the correct status of the system and the difference in the case of failure. These differences may help you to quickly and easily determine the cause of any problems when they happen.

1.2 Problem determination basics

Problem determination is the processes of finding what the problem is and why it happened.

- ▶ What is the problem?
 - Understand what the problem is, and obtain a consensus with the team (both IBMers and customers).
 - Gather the correct information related to the failure components.
- ▶ Why did the problem happen?
 - Specify the cause or trigger of the problem with supporting information (logs, traces, and so on).
 - Determine the cause, step-by-step, and collect information in various patterns.
 - Determine the most effective process to reproduce the problem, and find the difference between a problem that is reproducible and one that is not reproducible.

These need the correct supporting information related to the problematic component. Sometimes you need to reproduce and gather more information. Customers require recovery of their system as soon as possible. However, recovered systems will not provide effective information. A quick response and gathering of the correct information is very important. Information collected at the correct time and organized properly makes analysis faster.

1.2.1 Basic problem determination steps

Just after the problem happens, you must understand what the problem is and its impact to the customer, gather needed information, and decide how to analyze this information. The priority of analysis must be decided. Of course, recovery is

always a top priority and avoidance is second, but sometimes these cannot be done without first determining the cause.

The analysis phase should include exploring the cause and the solution. You should try to reproduce the problem. Problem reproduction may show the cause of the problem directly, or provide some big hints as to what it is. It will also make gathering additional information easier.

Figure 1-4 shows some basic steps of problem determination. Sometimes it is necessary to escalate to each support team at certain points in the analysis, as shown in Figure 1-4. Report (as shown in the figure) not only reports the result, but also includes discussion about follow-up actions.

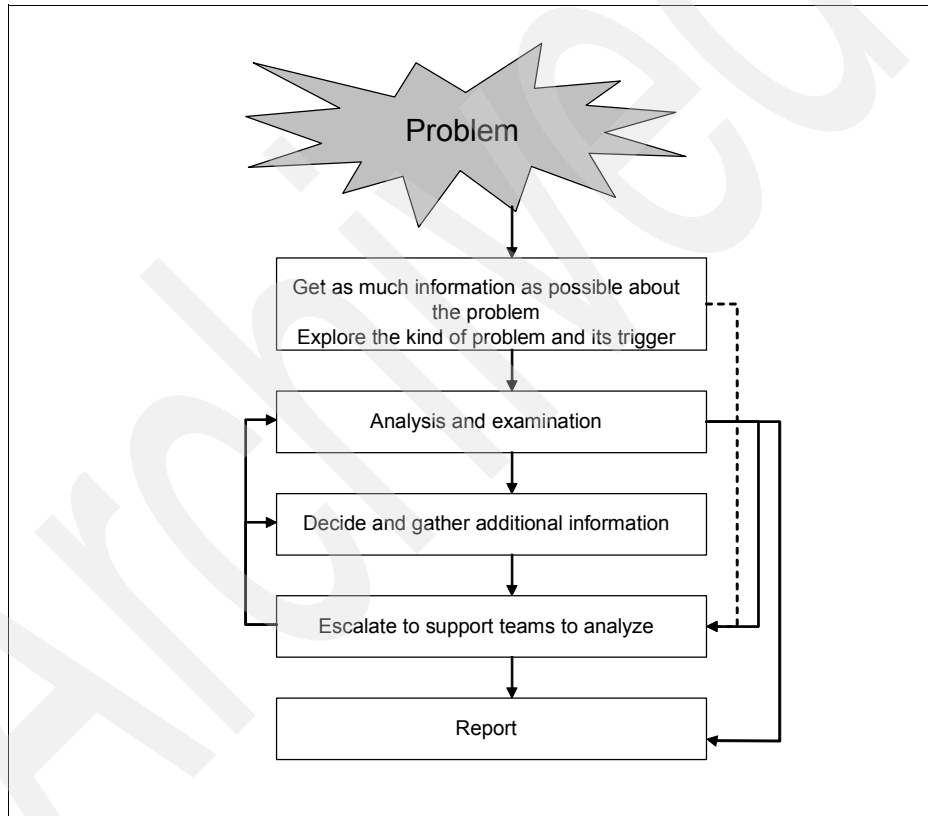


Figure 1-4 Basic steps of problem determination

Determine which component is failing in the order shown in Figure 1-5. First look at the application or middleware, and then look at the actual hardware to determine where the failure is originating.

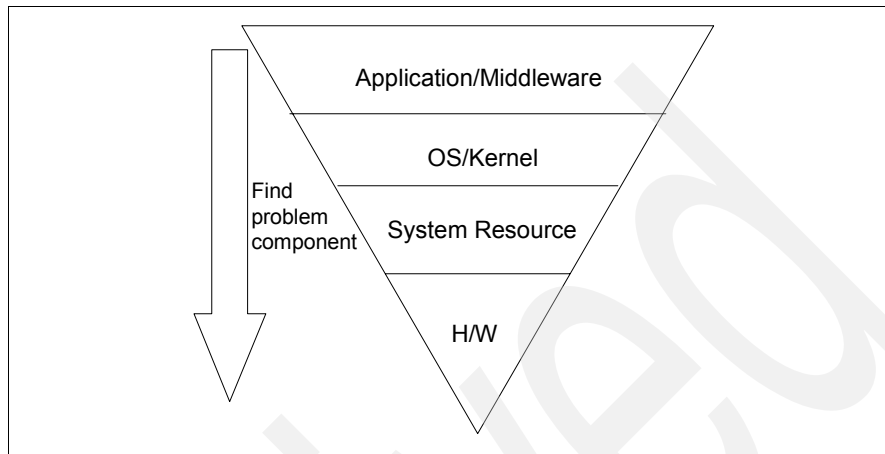


Figure 1-5 Find the problem component

1.2.2 Gather information and explore the type of problem and its trigger

Just after a problem happens, you should gather and organize problem determination data as soon as possible. You should reach a consensus among parties concerned as to what data to collect. This is the first step of problem determination. Often the cause of the problem can be found during this step.

Important: As soon as any problem occurs, execute `dbginfo.sh`. If Linux is down or hung, get a stand-alone dump.

Information about the problem

Collect as much information as possible about the problem, for example:

- ▶ A complete description of the problem
- ▶ Date and time of the problem
- ▶ Which system the problem appeared on and whether it appears on more than one system
- ▶ System environment - hardware, software (with patch level), network

- ▶ Other work occurring around you

Were other people performing work around you that may have caused the problem (such as a network maintenance outage)?
- ▶ What made you decide it was a problem (effect/logs)

A command or script log should be referenced.
- ▶ Current status (ongoing problem, recovered, or avoided)

Is there is a recovery plan or a plan for avoidance of the problem?
- ▶ Scope (multi-system or non-multi-function) - common terms or differences
- ▶ Difference from daily operation (loading, operation order, and so on)
- ▶ Past results
 - Has the target operation had a similar result or is this a new behavior?
 - Is there are any change from the previous results, or are they exactly the same?
- ▶ Same/similar problem

Has the same or a similar problem happened before?

 - If it has happened before, check the necessary information, the avoidance method, and the recovery procedure.
 - This has happened in the past and has already been solved, can we fix the problem to prevent it from happening again?

Impact

The impact of a problem on the following should be determined:

- ▶ Applications

For production/for development/in implementation
- ▶ Business
 - The importance of the service of the system
 - Status (recovered, partial failure, or full failure)

Information that should be gathered

The following is a list of information that you should collect about a problem to assist you in determining what the problem is:

- ▶ Define the symptoms of the problem (that is, boot failure, disconnection network, I/O failure)

- ▶ Find the error component (hardware, kernel, application, and so on).
We may have a chance to find the error component quickly based on the symptoms, error messages, or other signs.
- ▶ Confirm the current system status.
We can gather more information before recovery than after. If data is or wrapped or only in memory, reboot may clear it.
- ▶ Current system topologies (hardware, software, network).
- ▶ Decide what information should be collected.
At first, we should gather common information for any problems. If it is difficult to determine what you must collect, contact your support team.

Investigation policy

Determine an investigation policy with your team members. You will need to:

- ▶ Agree on which contents will be collected ahead of time and when that collection must take place.
- ▶ Agree on the type of recovery, avoidance, cause, and so on.
- ▶ Schedule a practice run of data collection.

1.2.3 Analysis of a problem

After you have gathered the needed information regarding the problem, you must examine and analyze this data to:

- ▶ Determine the cause of the problem.
- ▶ Determine a method to reproduce the problem.
- ▶ Determine an avoidance method (or work around) for the problem.

Tip: Confirm the validity of collected data before starting an analysis. Packed information should be unpacked, and it should be determined if it can be read or whether it must be formatted with tools in order to be read.

Determine the cause of the problem

Before you can decide on how to reproduce the problem or work around the problem, you are going to have to determine the cause of the problem. Some methods of determining the cause of the problem are:

- ▶ Specify the trouble component according to the problem description and gathered information.
- ▶ Search for information about the problem on Bugzilla (<http://www.bugzilla.org>) or your particular hardware and software

technical library. For Linux trouble, you can also find information about your problem by doing a Web search.

- ▶ Investigate the cause on each component.
- ▶ Determine whether you need to contact a support team.

Determine a method to reproduce the problem

Being able to reproduce a problem is an important aspect of both problem determination and problem resolution. To reproduce a problem, you must:

1. Determine what triggered the problem.
 - Was it a particular command?
 - Was it a particular transaction?
 - Was it a particular operation?
 - Does it occur during system start, stop, or reboot?
 - Were any changes recently applied to the system?
2. Check the conditions surrounding the problem.
 - Check resource use such as CPU, memory, I/O, and network.
 - Check other transactions or processes running in parallel.
3. Prepare the test system.
 - Use the test system.
 - Use the production system temporarily.
 - Confirm the effects on the system by reproducing the problem.
4. Test with related conditions and triggers.

If a problem is not reproducible, see 1.2.6, “How to approach complex problems” on page 16.

Determine an avoidance method

An avoidance method is a temporary solution or work around for a problem (for example, temporarily changing the operation or modifying the program or shell script). Basically, it avoids the trigger of the problem. Quick recovery procedures are also a part of avoidance methods if a problem has already been found.

Accepting the avoidance method may have some risks and cause additional workload. You need to discuss this option with your customers. The effects of changing operations and workloads by changing programs or scripts must be defined.

Note: Sometimes the work around could be the actual solution.

Gather additional information

Sometimes you may need to investigate a problem step-by-step with additional information. We need to define what information is needed according to the current investigation status. When analyzing a problem, it may be determined that more information is needed.

1.2.4 Collect additional information

Collecting additional information about the customer's environment or a similar test environment may be necessary after an analysis of data shows that you do not have enough information about a problem. You may need a consensus with your team to determine the current investigation status and check the effects on the system of gathering more information.

Consensus terms

Your team will have to come to a consensus on the following issues before you can collect additional information:

- ▶ Was anything found regarding the problem using the current information?
- ▶ What other information is required to continue the problem investigation?
- ▶ What needs to be done to collect the additional information?
 - Do you need to reproduce the problem to collect additional data? See “Determine a method to reproduce the problem” on page 14. Run a test and gather the additional information.
 - Set tools for gathering information in daily operation (effects of settings on production).
- ▶ How should you gather additional data?
 - Run a reproduction test to see whether the problem recurs.
 - Set tools for gathering information in the system if the problem is not found.

Effects

You must understand the importance of collecting additional information as well as the effects this gathering of information will have on a system.

- ▶ You may need exclusive use of the system to run the reproduction test.
 - Schedule to use the system.

- Take into consideration the additional system load of the test and the time that it takes to gather the information.
- ▶ You may need to change the settings of some of your tools in the system with the effects on the system of:
 - System additional loading (overhead)
 - Data size (trace, log, and so on)
 - Effects on operation
 - Changed scope of operation, configuration, script
 - Problem detection method during problem reproduction

Note: Gathering additional information sometimes overloads the system. Make sure that all persons involved in the upkeep of the system understand the implications and importance of gathering additional information.

1.2.5 Escalate to the support team to analyze

If you are unable to find the cause of the problem, you may need to contact a support team for further analysis. The team will ask you to provide additional information. The information that they may need is:

- ▶ Frequency
 - Date when the problem happened
 - How often the problem happened
- ▶ Trigger (Is there something that happens first before the problem?)
 - What was happening at the time the problem occurred?
 - Was there recently a change to the system?
- ▶ Reproducibility

Reproduction procedure.
- ▶ Required response

Investigate the cause, the avoidance method, and the recovery method and their priority.

1.2.6 How to approach complex problems

Problem reproduction helps in determining the exact cause of the problem. If the problem is not reproducible, it is difficult to find the cause. It is important to gather enough and accurate information.

Ideally, you should try to reproduce the problem in a test environment. Automated and continual operation around the trigger is a good place to start if you know the trigger. Automated test systems allow you to monitor the problem occurring and stop the test if a problem occurs. Automated collection of data is also helpful.

Your team should decide how long testing should continue and prepare different types of test cases. If one type does not reproduce the problem for a specified amount of time, change to another type of test case.

In our sample test case, LVM command hangs. In this case you should:

1. Prepare a Linux test system that has the same version of kernel, LVM, and DM.
2. Create a script that runs the LVM commands.
It does not include “&” for the command, so it stops if the LVM command hangs.
3. Set core dump.
4. Run the script with two patterns:
 - Singly
 - Parallel
5. The problem is reproduced when script runs in parallel. This demonstrates that the problem is an LVM specification for that version. Change the operation to not run the LVM command in parallel to ensure that your understanding of the problem is correct.

Set tools to gather information

The tools that you use to collect data should include dumps, EREP reports, and traces. Getting a stand-alone dump will have impact to services, so ensure that you communicate your needs and the necessity to your team.

If the trouble component is not found, set monitor tools for all related components, not only for gathering information (application, middleware, OS, system resource, performance), and if the problem recurs, collect the data that is related to the problem.

1.3 Problem resolution and follow-up actions

After determining the problem you must fix the problem. Applying a patch or changing a configuration and operation may cause side effects. Before applying patches or making configuration changes, confirm that the problem will be fixed

with it. Confirm that there will be no side effects. Ensure that you update your documentation.

Before applying a fix (patch or change), we recommend doing a pre-apply test to confirm that the problem will be fixed with no side effects. This test should include a test to try and reproduce the problem. If the problem did not happen frequently, you need to determine how long to wait for the problem to recur to ensure that you have truly fixed it.

For applying a patch or program modification you will need to test it in the production system or a test system that is similar to your production system. To confirm problem resolution, a reproduction test should show a clear result.

1.3.1 Expanded check for other systems

Check all of your other systems to make sure that the same problem does not happen. Systems with the same version, configuration, and operation may have the same problem.

The following list is a reminder of what to watch for on your other systems:

- ▶ Problem conditions. Check to see whether there are any other conditions that cause the same problem. Check the:
 - Software version
 - Transaction type
 - Operation
 - Configuration
- ▶ System scan
Scan all systems that have the possibility of having the same condition.
- ▶ Schedule for applying fixes.
 - If the possibility of a system having the same problem is high, apply the fix as soon as possible
 - If the possibility is low, schedule the application of the fix for the next maintenance cycle or determine whether applying no fix at all is an option.

1.3.2 Update

After the change is applied to the system, you need to remember to update your documents, monitoring procedures, and problem determination operations to reflect this new problem.

This documentation may include your message and code manuals. If applying a fix makes a difference to all of the same systems, you should include it in the document updates. Loss of that information may cause other problems and make problem determination difficult.

Verify that applying a fix does not have any new messages. If there are new messages, such as that there is additional system loading, make sure that you document and discuss any new threshold values for the system.

Additionally, update for:

- ▶ New message or changing monitored message
 - Add or change the target message (TEC, Console, AOEMF, IBM Director, and so on).
 - Add or change messages and code manuals.
- ▶ Additional system loading
 - Change the threshold value.
 - Add new monitoring contents.

You may need to change from monitoring content that is no longer necessary to monitoring new content.

After confirmation of the problem resolution, review your problem determination operations and update your procedures accordingly. You may need to update your procedures for problem determination. Some of the updates that you may need to make are:

- ▶ System environment
 - Should you run reproduction tests?
 - Did monitoring systems find the problem?
 - Are the systems automated enough?
- ▶ Actions just after problem happens
 - Are prepared operation manuals correct for this problem?
 - Do operators know the correct procedure?
 - Is there any time loss for first actions?
- ▶ Gathering information
 - Are you missing any problem determination information?
 - Are there enough resources for gathering information?
 - Operation personnel
 - Automation scripts

- System resources
- Reporting
 - Do all reports have the correct information related on each report line?
 - Lack of information
 - Unnecessary information
 - Is reporting time correct?
 - Is contact timing for the support team correct?

1.4 Summary

In this chapter, we described a methodology for determining the cause of a problem on your Linux on System z environment. The first step in finding the cause of a problem so that you can correct it is to gather as much information as possible about the circumstances surrounding the problem. Write as much information down about the problem as possible. Examine whether the problem occurs on just one system or on many of your systems. Determine whether a change you or somebody else has made to the system happened just before the problem started.

Additionally, examine the frequency of the problem occurring and whether it is a problem that can be reproduced. You must have a test system that is set up as close to the production system as possible so that you can try to reproduce the problem there.

In this rest of this book we continue to examine problem determination methods and the tools on z/VM as well as Linux on System z that can assist you in finding the cause of your problem quickly to bring about a quick resolution.

Problem determination tools for z/VM

In this chapter we show the tools that are available for problem determination and data gathering in the z/VM environment. There are tools available on both Linux and z/VM for problem determination. Dependent on the problem area or category, you need to decide which tools should be used to help determine your problem.

Since Linux runs as a guest to z/VM, Linux-based tools have some degree of error during problem determination in areas where the resources are fully controlled by z/VM. At this point, we recommend that you use z/VM-based tools or a combination of z/VM and Linux-based tools to get more granular information about the problem. On the other hand, Linux-based tools are often your best choices for debugging an application, and kernel-related or process-related problems.

First we discuss some z/VM system tools. We provide some CP commands that are available to assist you with gathering some basic information about the status of the system.

Additionally, we show the features of IBM z/VM Performance Toolkit and demonstrate how to use it for problem determination. We also provide an overview of the functions of Omegamon.

2.1 z/VM system tools

Like Linux, the z/VM operating system has commands to display system information as well as system performance information. These commands are control program (CP) system commands. In this section we provide some CP commands that you can use to gather basic information about the status of the system.

2.1.1 Collecting technical setup information

Collect technical setup information as part of your information-gathering technique. To gather your release-level and service-level information use the command `q cplevel`, as shown in Figure 2-1.

```
q cplevel
z/VM Version 5 Release 3.0, service level 0703 (64-bit)
Generated at 03/13/08 07:41:44 EDT
IPL at 03/24/08 14:25:10 EDT
```

Figure 2-1 Sample output from the `q cplevel` command

Other commands that help you collect information about the setup of your environment are:

- ▶ Network setup
 - **q lan**
 - **q nic**
 - **q vswitch**
 - **q v osa**
- ▶ General and DASD
 - **q set** (See Figure 2-2.)
 - **q v dasd**

```
q set
MSG ON , WNG ON , EMSG ON , ACNT OFF, RUN OFF
LINEDIT ON , TIMER OFF , ISAM OFF, ECMODE ON
ASSIST OFF , PAGEX OFF, AUTOPOLL OFF
IMSG ON , SMSG OFF , AFFINITY NONE , NOTRAN OFF
VMSAVE OFF, 370E OFF
STBYPASS OFF , STMULTI OFF 00/000
MIH OFF , VMCONIO OFF , CPCONIO OFF , SVCACCL OFF , CONCEAL OFF
MACHINE XA , SVC76 CP, NOPDATA OFF, IOASSIST OFF
CCWTRAN ON, 370ACCOM OFF, TIMEBOMB IDLE
```

Figure 2-2 Output from *q set* command

2.1.2 Indicate

A good source of information about the overall status of the z/VM system is the **indicate** command. The response to the **indicate** command varies depending on the CP class of the virtual machine on which the command was issued. We assume for the examples in this book that you issue the commands from a virtual machine with the appropriate CP classes like the MAINT user ID.

Use **indicate load** to display the operating load on the host system:

- ▶ The percentage of usage and CPU type for each online processor in your system
- ▶ Information concerning expanded storage
- ▶ Information concerning the minidisk cache
- ▶ The paging rate
- ▶ The number of users in the dispatch, eligible, and dormant lists

See the response to the **indicate** command in Figure 2-3.

Note that **Load** is the default operand of the **indicate** command.

```
indicate load
AVGPROC-000% 04
XSTORE-000004/SEC MIGRATE-0000/SEC
MDC READS-000001/SEC WRITES-000001/SEC HIT RATIO-100%
PAGING-2/SEC STEAL-000%
Q0-00001(00000)                                DORMANT-00022
Q1-00001(00000)                                E1-00000(00000)
Q2-00000(00000) EXPAN-001 E2-00000(00000)
Q3-00002(00000) EXPAN-001 E3-00000(00000)

PROC 0000-001% CP      PROC 0001-000% CP
PROC 0002-000% CP      PROC 0003-000% CP

LIMITED-00000
Ready; T=0.01/0.01 11:38:16

RUNNING  VMLINUX8
```

Figure 2-3 *indicate load command*

Use **indicate paging** to do the following:

- ▶ Display a list of the virtual machines in page wait status.
- ▶ Display page residency data for all system users.

Use the command with the **wait** operand to identify the virtual machines that are currently in page wait and that displays the number of pages allocated on auxiliary storage and expanded storage for each one. This is the default.

Use **paging wait** when the **indicate queues** command has shown that a significant proportion of the active users of the system are persistently in page wait.

When issued with the **all** operand, the command displays the page residency data for all users of the system. See Figure 2-4 for an example of the **indicate paging wait** command.

```
indicate paging wait
No users in page wait
Ready; T=0.01/0.01 15:57:33
indicate paging all
MAINT      000000:001396 TEACHOR2 008162:054438 LXORINST 001847:010210
TEACHOR1   020713:241153 LNXMAINT 000000:000175 LNXINST  022184:080382
FTPSEVER   000000:001412 TCPPIP   000226:002578 USER15   000000:000875
TCPMAINT   000000:000200 USER1    000000:000887 EREP      000000:001233
OPERSYMP   000000:001266 DISKACNT 000003:001227 LOGNSYSC 000000:000000
DTCVSW2    000004:002518 DTCVSW1 000000:002566 VMSERV   000000:001181
VMSERVU    000000:001181 VMSERVS 000000:001326 OPERATOR 000000:000001
LXOR4      000000:000000 LXOR1    000000:000001 PERFSVM  000000:000000
LXOR3      002022:000001 USER14   000000:000901
Ready; T=0.01/0.01 15:57:41
```

RUNNING VMLINUX8

Figure 2-4 Indicate paging command

Use the **indicate queues** command (see Figure 2-5) to display, in order of their priority, current members of the dispatch and eligible lists. Issued with the **exp** operand, it displays expanded information about users with a transaction in progress.

```
indicate queues
MAINT      Q1 R00 00000686/00000666 LXORINST      Q1 PS 00515332/00515287
LXOR1      Q1 PS 00137239/00137193 LXOR3          Q3 PS 00115756/00138801
TCPIP      Q0 PS 00001026/00000447
Ready; T=0.01/0.01 16:02:32
indicate queues exp
MAINT      Q1 R00 00000686/00000666 .I.. .0005 A00
LXOR3      Q3 PS 00115756/00138801 ..D. 99999 A02
TCPIP      Q0 PS 00001026/00000447 .I.. 99999 A03
Ready; T=0.01/0.01 16:02:34
```

RUNNING VMLINUX8

Figure 2-5 Indicate queues command

The first column following the user ID indicates the list status of the user ID:

- ▶ Q0: The dispatch list and exempt from eligible list delays
- ▶ Q1: The dispatch list and entered as an E1 user
- ▶ Q2: The dispatch list and entered as an E2 user
- ▶ Q3: The dispatch list and entered as an E3 user
- ▶ E1: The E1 eligible list
- ▶ E2: The E2 eligible list
- ▶ E3: The E3 eligible list
- ▶ L0: The limit list and exempt from eligible list delays
- ▶ L1: The limit list and the user entered the dispatch list as an E1 user
- ▶ L2: The limit list and the user entered the dispatch list as an E2 user
- ▶ L3: The limit list and the user entered the dispatch list as an E3 user

More information about the **indicate** command and how to interpret the response to that command can be found in the manual *z/VM CP Commands and Utilities Reference*, SC24-6081-05.

2.1.3 query srm

For an overview of the system-wide parameters used by the CP scheduler we use the **query srm** command. Some problems may be caused by inappropriate settings of this system resource parameters. In Figure 2-6 you see the response of the **query srm** command.

```
q srm
IABIAS : INTENSITY=90%; DURATION=2
LDUBUF : Q1=100% Q2=75% Q3=60%
STORBUF: Q1=125% Q2=105% Q3=95%
DSPBUF : Q1=32767 Q2=32767 Q3=32767
DISPATCHING MINOR TIMESLICE = 5 MS
MAXWSS : LIMIT=9999%
..... : PAGES=999999
XSTORE : 0%
Ready; T=0.01/0.01 11:42:45
```

RUNNING VMLINUX8

Figure 2-6 query srm command

For more detailed information about how to set and adjust the srm parameters see *Linux on IBM System z: Performance Measurement and Tuning*, SG24-69261.

2.1.4 query alloc page

To check the amount of paging space that is available and used by the system, use the **query alloc page** command. A high percentage of used paging space over a log period of time can cause performance problems and may also cause an ABEND if page space is no longer available.

Note: z/VM warns about page space problems when approximately 90% of the paging space is used.

See Figure 2-7 for an example of a **query alloc page** command.

```

q alloc page

```

VOLID	RDEV	EXTENT START	EXTENT END	TOTAL PAGES	PAGES IN USE	HIGH PAGE	% USED
LX8PAG	CF33	1	3338	600840	280359	597573	46%
SUMMARY				600840	280359		46%
USABLE				600840	280359		46%

Ready; T=0.01/0.01 13:45:38

RUNNING

VMLINUX8

Figure 2-7 Query alloc page command

2.1.5 VM trace

As described in 3.2.2, “When a trace should be run” on page 82, sometimes it is better to start a trace instead of generating a dump. You may be asked by IBM

Software Support to run a trace in order to investigate a problem. We describe here how to do so and provide an example. Interpreting this trace is beyond the scope of this book, but you will need to know how to generate a trace when IBM Software Services asks you to produce a trace to investigate a problem. To set up and control the execution of a trace within z/VM we can use the CP command **trace**.

The virtual machine tracing facility allows you to follow or trace the execution of almost anything that takes place while you are using your virtual machine. For example, you can trace instructions, I/O interrupts, branches, and changes to storage. You can limit tracing to a specified storage range and to events that affect a specific device.

Each traced event results in a trace entry, a command response that you can have sent to your virtual console, to a virtual printer, or to both. The trace entry is made up of significant information about the event. You can use trace entries to analyze the operation of your virtual machine and to debug problems.

Before you can use the **trace** command, you must know the meaning of a *trace trap* and *trace sets*.

- ▶ A trace trap is a unit of control that allows you to tell CP what event you want to trace and the conditions under which you want the event traced.
- ▶ A trace set is a collection of one or more trace traps that are active at the same time. To trace different types of events, you can pass control between different trace sets. A trace set can also contain no trace traps, in which case the trace set is called a null trace set. You can use null trace sets to temporarily stop tracing.

After you have decided to use trace:

1. Choose events that you want to trace.
2. Name a trace set and create the appropriate traps.
3. Run the program that you want to trace. You can trace anything that runs in your virtual machine. Modify your existing trace sets or create new sets as needed.

In our example we want to trace all instructions that occur between storage locations 40000 and 41000. We do not want CP to stop the virtual machine when it traps an event. The command to set up the trace trap is shown in Example 2-1. Since the trace could produce many lines of messages, we should be careful to collect all of them for further detailed investigation. Therefore, we spooled the console messages with the command shown in bold on the second line in Example 2-1. Subsequently, we checked the setting of the trace trap with the **q trace** command.

Example 2-1 Example of the trace command

```
#cp trace instruction from 40000-41000 run
#cp spool cons * start
#cp q trace
```

NAME	INITIAL	(ACTIVE)
1	INSTR	PSWA 00040000-00041000
	TERM	NOPRINT RUN SIM
	SKIP 00000	PASS 00000 STOP 00000 STEP 00000
	CMD	NONE

The trace is now active, and we issue the commands shown in Figure 2-8. We load the **vmcp** module, which allows us to send commands from the SSH session of our Linux system to CP. Then we issued two **cp query** commands using the **vmcp** module.

```
[root@user12 /]# modprobe vmcp
[root@user12 /]# vmcp q n
TEACHOR1 - DSC , USER13 -L0004, LXOR2 -L0008, PERFSVM - DSC
OPERATOR - DSC , TEACHOR2 - DSC , LXOR3 - DSC , LXORINST - DSC
LXOR4 - DSC , LNXINST - DSC , TCPIP - DSC , DTCVSW2 - DSC
DTCVSW1 - DSC , VMSERVER - DSC , VMSERVU - DSC , VMSERVS - DSC
OPERSYMP - DSC , DISKACNT - DSC , EREP - DSC , USER12 -L0003
VSM - TCPIP
[root@user12 /]# vmcp q dasd
DASD 0111 9336 (VDSK) R/W 512000 BLK ON DASD VDSK SUBCHANNEL = 0003
DASD 0190 3390 LX8RES R/O 107 CYL ON DASD CF31 SUBCHANNEL = 000C
DASD 0191 3390 LX8UR1 R/W 10 CYL ON DASD CF36 SUBCHANNEL = 0004
DASD 019D 3390 LX8W01 R/O 146 CYL ON DASD CF34 SUBCHANNEL = 000D
DASD 019E 3390 LX8W01 R/O 250 CYL ON DASD CF34 SUBCHANNEL = 000E
DASD 0291 3390 LX8UR2 R/W 350 CYL ON DASD CF37 SUBCHANNEL = 0005
DASD 0300 3390 LXD43C R/W 10016 CYL ON DASD D43C SUBCHANNEL = 0006
DASD 0400 3390 LXD43D R/W 10016 CYL ON DASD D43D SUBCHANNEL = 0007
DASD 0401 3390 LX8W01 R/O 146 CYL ON DASD CF34 SUBCHANNEL = 0010
DASD 0402 3390 LX8W01 R/O 146 CYL ON DASD CF34 SUBCHANNEL = 000F
DASD 0405 3390 LX8W01 R/O 156 CYL ON DASD CF34 SUBCHANNEL = 0011
[root@user12 /]#
```

Figure 2-8 Linux commands to generate trace messages

The trace messages are displayed in parallel on the 3270 console. See Figure 2-9.

0000000000040FFC'	LGR	B9040032	0000000000000015		
			0000000000000015	CC 2	
0000000000041000'	LG	E3200DD80004	000000000000DD8	CC 2	
-> 0000000000040FA4'	STMG	EBCFF0780024	>> 000000000747BC48	CC 2	
0000000000040FAA'	TMLL	A7F13F00	CC 2		
0000000000040FAE'	LGR	B90400EF	000000000747BBD0		
			000000000747BBD0	CC 2	
0000000000040FB2'	BRZ	A7840001	0000000000040FB4	CC 2	
0000000000040FB6'	AGHI	A7FBFFD8	CC 2		
0000000000040FBA'	LARL	C010001ADAA3	CC 2		
0000000000040FC0'	LGR	B90400C2	000000001E866BE8		
			000000001E866BE8	CC 2	
0000000000040FC4'	STG	E3E0F0980024	>> 000000000747BC40	CC 2	
0000000000040FCA'	LG	E31010000004	????????????????	CC 2	
0000000000040FD0'	LG	E31010200004	????????????????	CC 2	
0000000000040FD6'	BASR	0DE1	-> 000000000011C588'	CC 2	
0000000000040FD8'	LTR	1222	CC 0		
0000000000040FDA'	LHI	A7280000	CC 0		
0000000000040FDE'	BRNZ	A7740006	0000000000040FEA	CC 0	
0000000000040FE2'	OI	9601C01E	>> 000000001E866C06	CC 1	
0000000000040FE6'	LHI	A7280001	CC 1		
0000000000040FEA'	LGFR	B9140022	0000000000000001		
					HOLDING VMLINUX8

Figure 2-9 Trace message on the 3270 console

When you have collected enough trace messages you can stop the trace. As mentioned earlier, we spooled the console messages that are displayed on the 3270 console to have it available for further analysis. Now we need to stop the console spooling. The commands to do this are shown in Figure 2-10. With the **cp close console** command the console messages are collected and stored as a reader file. Now this reader file can be received and stored as a CMS file for analysis or sent to another system, or both.

```
CP TRACE END
Trace ended

CP CLOSE CONS
RDR FILE 0046 SENT FROM USER12    CON WAS 0046 RECS 058K CPY 001 T NOHOLD NOKEEP

CP Q RDR ALL
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME      NAME      TYPE      DIST
USER12   0045 V DMP 00048723 001 USER 03/24 14:17:32 VMDUMP    FILE      USER12
USER12   0046 T CON 00057686 001 NONE 03/25 15:56:39                USER12
```

RUNNING VMLINUX8

Figure 2-10 Stop the trace

For more detailed information about the features of the **trace** command see the manuals *z/VM CP Commands and Utilities Reference*, SC24-6081-05, and *z/VM Virtual Machine Operation*, SC24-6128-02.

2.1.6 vmdump

In the case of your Linux guest system having crashed, generate a dump of the storage for further problem determination. For this purpose we use the CP command **vmdump**.

First we need a minidisk large enough to store the dump. In our example we used a virtual disk (VDISK) in storage with 1005000 blocks. We choose to access the VDISK as filemode B. The commands to do this are shown in Figure 2-11.

```
def vfb-512 as 999 blk 1005000
DASD 0999 DEFINED
Ready; T=0.01/0.01 15:04:42
format 999 b
DMSFOR603R FORMAT will erase all files on disk B(999). Do you wish to continue?
Enter 1 (YES) or 0 (NO).
1
DMSFOR605R Enter disk label:
temp
DMSFOR733I Formatting disk B
DMSFOR732I 1005000 FB-512 blocks formatted on B(999)
Ready; T=0.01/0.01 15:04:50
```

RUNNING VMLINUX6

Figure 2-11 *Preparing a minidisk to store the dump*

We have now enabled the minidisk to store the dump. To force the system into a kernel panic we used a kernel module that was written for this purpose. In Figure 2-12 we see on the 3270 console of our Linux system, LNXIHS, the kernel panic messages. Now the **vmdump** command is issued and a dump of the storage is generated and stored as a reader file with the spoolid 0017.

```
kpanic: module not supported by Novell, setting U taint flag.
kpanic: module license 'unspecified' taints kernel.
ITS0 Kernel Panic test.
Invoking a Kernel Panic .. :-(
Kernel panic - not syncing: Panic
HCPGIR450W CP entered; disabled wait PSW 00020001 80000000 00000000 208BC068

cp vmdump
Command complete

cp q rdr all
ORIGINID FILE CLASS RECORDS CPY HOLD DATE TIME NAME TYPE DIST
LNXIHS 0017 V DMP 00044503 001 NONE 03/24 15:10:24 VMDUMP FILE LNXIHS
LNXIHS 0013 T CON 00000338 001 NONE 09/28 07:57:51 LNXIHS
LNXIHS 0015 T CON 00000474 001 NONE 03/12 08:11:39 LNXIHS
LNXIHS 0014 T CON 00000900 001 NONE 12/08 11:41:29 LNXIHS
```

CP READ VMLINUX6

Figure 2-12 Kernel panic, generating the dump using the vmdump command

To transfer the dump into a form that allows for further problem determination, we need to process the dump. This means that first we must load the dump from the reader spool file and then store it as a CMS dump file. To do this, use the CP facility **dumpload**. In Figure 2-13 you see the execution of the dumpload facility and the corresponding system messages. When the dump is stored as a reader file and you want to have the output generated by dumpload stored on the a-disk, you can issue the command without any options. In our example we want to store the processed dump file on the minidisk accessed in filemode B.

```
dumpload spool outfile prb00000 b
HCPEDZ8183I DUMLOAD z/VM VERSION 5 RELEASE 3.0
HCPEDZ8150I PROCESSING DUMP TO PRB00000 DUMP0001 B
HCPEDZ8167I VIRTUAL MACHINE DUMP FROM z/VM V05R03M0
HCPEDZ8168I VIRTUAL MACHINE DUMP, FORMAT=FILE,
DUMPID=
HCPEDZ8156A DO YOU WANT TO PROCESS THIS DUMP? (YES/NO)
yes
Ready; T=0.23/1.08 15:15:49
```

RUNNING VMLINUX6

Figure 2-13 Dumpload utility

Subsequently, we transferred the processed dump file from the minidisk accessed in filemode B to another system for further investigation. The transfer was done using FTP. In Figure 2-14 you see this transfer with all necessary commands and the corresponding system messages.

```
ftp 9.12.4.244
VM TCP/IP FTP Level 530
Connecting to 9.12.4.244, port 21
220 (vsFTPD 2.0.4)
USER (identify yourself to the host):
root
>>>USER root
331 Please specify the password.
Password:

>>>PASS *****
230 Login successful.
Command:
bin
>>>TYPE i
200 Switching to Binary mode.
Command:
put prb00000.dump0001.b dplnxihs
>>>SITE VARrecfm
500 Unknown SITE command.
>>>PORT 9,12,4,89,4,4
200 PORT command successful. Consider using PASV.
>>>STOR dplnxihs
150 Ok to send data.
226 File receive OK.
182284288 bytes transferred in 3.671 seconds. Transfer rate 49655.21 Kbytes/sec.
Command:
quit
>>>QUIT
221 Goodbye.
Ready; T=0.33/0.73 15:24:45
```

RUNNING VMLINUX6

Figure 2-14 Transfer the processed dump file to another system via FTP

For more detailed information about the **vmdump** command and the **dumpload** CP facility, see the manual *z/VM CP Commands and Utilities Reference*, SC24-6081-05.

2.2 IBM z/VM Performance Toolkit

The IBM z/VM Performance Toolkit (also called the Performance Toolkit) is designed to assist operators and system programmers or analysts in the following areas:

- ▶ System console operation in full panel mode
The features provided have been designed to facilitate the operation of VM systems, thereby improving operator efficiency and productivity.
- ▶ Performance monitoring on z/VM systems
An enhanced real-time performance monitor allows system programmers to monitor system performance and to analyze bottlenecks. Features included in the monitor, and the manner in which data is displayed, have both been designed to improve the system programmer's productivity when analyzing the system, and to allow even a new user to work efficiently with the tool. The Performance Toolkit can help system programmers to make more efficient use of system resources, increase system productivity, and improve user satisfaction.

The IBM z/VM Performance Toolkit features include:

- ▶ Automatic threshold monitoring of many key performance indicators with operator notification if user-defined limits are exceeded
- ▶ Special performance monitoring mode with displays for monitoring:
 - General CPU performance
 - System and user storage utilization and management
 - Channel and I/O device performance, including cache 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 - system execution space (SXS) performance data
- Virtual networking configuration and performance data

The collected performance data is also used to provide:

- ▶ A re-display facility for many of the key figures shown on the general CPU performance and storage utilization panels, which allows browsing through the last measurements (up to twelve hours)
- ▶ Graphical history plots with a selection of up to four re-display variables, either as simple plots or, if the Graphical Data Display Manager program (GDDM®, 5684-007 or 5684-168) is available, also in the form of GDDM graphics
- ▶ Graphical variable correlation plots (simple plots or GDDM graphics) that show how the values of specific variables are correlated to the values of another variable

Note: GDDM is required for generating graphics on the console of a virtual machine. However, no additional software is required when generating graphics with Web browsers using the WWW interface.

For future reference, the accumulated performance data can also be used for creating:

- ▶ Printed performance reports
- ▶ Simple performance history files
- ▶ Extended trend files

For capacity planning, the history files on disk can be used for:

- ▶ Performance trend graphics
- ▶ Numerical history data analysis

2.2.1 Modes of operation

The modes of operation for the Performance Toolkit include:

- ▶ Basic command mode (usually referred to as basic mode) is the normal operating mode that allows entering 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 about 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.
- ▶ The 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 (FCXTREND disk file).

IBM z/VM Performance Toolkit is a highly functional, varied coverage, highly detailed reporting and real-time monitoring tool. Communication with, and subsequent collection of, data from a Linux guest allows it to produce a consolidated view of the hardware, LPAR, VM, and Linux levels.

After startup you should see a window similar to Figure 2-15, with a header line at the top, an information message telling you about the maintenance level of the program, a command input line, and a bottom line that shows PF-key usage.

```
FCX001                      Performance Toolkit for VM                      Autoscroll 12
FCXBAS500I Performance Toolkit for VM FL530
FCXOMC772I SEGOUT data collection is active. Using segment: PERFOUT
FCXPMN446E Incomplete monitor data: SAMPLE CONFIG size too small
HCPMOF6229E Monitor event collection is already active.
HCPMOG6229E Monitor sample collection is already active.

Command ==>
F1=Help  F2=Redisplay  F3=Quit  F12=Return
```

Figure 2-15 IBM z/VM Performance Toolkit initial window

This initial window mode is described as basic mode in the documentation. It allows you to work in a manner not very different from native CMS. For our purpose to get information to help us with problem determination, we need to use the performance monitor mode of IBM z/VM Performance Toolkit. Enter the command MON or **monitor** to start working in performance monitor mode. Your window should look similar to Figure 2-16.

Performance Screen Selection (FL530)			Perf. Monitor
FCX124			
General System Data	I/O Data		History Data (by Time)
1. CPU load and trans.	11. Channel load		31. Graphics selection
2. Storage utilization	12. Control units		32. History data files*
3. Reserved	13. I/O device load*		33. Benchmark displays*
4. Priv. operations	14. CP owned disks*		34. Correlation coeff.
5. System counters	15. Cache extend. func.*		35. System summary*
6. CP IUCV services	16. DASD I/O assist		36. Auxiliary storage
7. SPOOL file display*	17. DASD seek distance*		37. CP communications*
8. LPAR data	18. I/O prior. queueing*		38. DASD load
9. Shared segments	19. I/O configuration		39. Minidisk cache*
A. Shared data spaces	1A. I/O config. changes		3A. Storage mgmt. data*
B. Virt. disks in stor.			3B. Proc. load & config*
C. Transact. statistics	User Data		3C. Logical part. load
D. Monitor data	21. User resource usage*		3D. Response time (all)*
E. Monitor settings	22. User paging load*		3E. RSK data menu*
F. System settings	23. User wait states*		3F. Scheduler queues
G. System configuration	24. User response time*		3G. Scheduler data
H. VM Resource Manager	25. Resources/transact.*		3H. SFS/BFS logs menu*
	26. User communication*		3I. System log
I. Exceptions	27. Multitasking users*		3K. TCP/IP data menu*
	28. User configuration*		3L. User communication
K. User defined data*	29. Linux systems*		3M. User wait states
<p>Pointers to related or more detailed performance data can be found on displays marked with an asterisk (*).</p>			
<p>Select performance screen with cursor and hit ENTER Command ==></p>			
<p>F1=Help F4=Top F5=Bot F7=Bkwd F8=Fwd F12=Return</p>			

Figure 2-16 IBM z/VM Performance Toolkit main screen

The user interface to IBM z/VM Performance Toolkit is menu driven, but you can fastpath by entering the full or abbreviated report name to go directly to the one

that you need. Once inside a report, many of them have sub menus, or further navigation is possible by placing the cursor under the variable and pressing the Enter key. Indeed, a lot of reports contain many layers of navigation, for example, you can look at an ordered list of DASD volumes and see the overall activity, in addition to controller functions. Then you can go deeper, and look at a given volume in more detail, then go down into further detail and look at the minidisk device activity. This type of navigation functionality greatly facilitates problem determination, as well as making the overall understanding of your system, even down at a deep level, much easier because of this enhanced accessibility. The various data reports are divided into the groups of general system data, I/O data, history data, and user data.

Performance screen items whose numbers do not appear highlighted on the initial menu cannot be selected because CP monitor data has not been collected for them. Possible reasons are:

- ▶ The required monitor records are not available because the CP level of your VM system does not yet produce them.
- ▶ The required monitor domains have not yet been enabled.
- ▶ The first monitor sample interval has not ended and we are still waiting for the initial set of monitor records.
- ▶ The display is intended for data that cannot be generated on the system (for example, LPAR data on an unpartitioned system).

2.2.2 Selecting performance screens

The screens are grouped into general system data, I/O data, user data (which shows data for either the last monitor sample interval or overall averages), and a series of by time log files that show a separate set of values for each monitor sample interval. They can be selected from the initial menu by taking one of the following actions:

- ▶ Entering the number of the selected performance data displayed on the command line
- ▶ Moving the cursor to the selected display (Note that it will move only to highlighted numbers if you use the tabulator keys.)
- ▶ Entering the '**short-path**' command for the desired screen

Then press the Enter key.

To get some meaningful data for problem determination you need know where to get this data within the performance toolkit.

A good starting point from which to gather information about the current status of the system are the screens in the category general system data. Starting with the first option, we begin with a performance health check of the system. Select 1 and press Enter or place the cursor into the appropriate field on the screen and press Enter. This option shows the overall processor utilization, system contention, and information about the user population. The user data includes the total number of users, number of active users, and number of users who are waiting for a resource like storage pages or I/O.

The screen displayed is shown in Figure 2-17.

FCX100	CPU 2094	SER 7991E	Interval 11:27:27 - 11:28:27	Perf. Monitor
CPU Load				Vector Facility Status or
PROC TYPE %CPU %CP %EMU %WT %SYS %SP %SIC %LOGLD %VTOT %VEMU REST ded. User				
P00 CP 12 1 12 88 1 0 92 12 Master				
P01 CP 29 0 29 71 0 0 81 29 Alternate				
P02 CP 34 0 34 66 0 0 87 34 Alternate				
P03 CP 23 0 23 77 0 0 81 23 Alternate				
Total SSCH/RSCH 122/s		Page rate 9.6/s		Priv. instruct. 263/s
Virtual I/O rate 116/s		XSTORE paging 804.9/s		Diagnose instr. 1/s
Total rel. SHARE 700		Tot. abs SHARE 0%		
Queue Statistics:	Q0	Q1	Q2	Q3
VMDBKs in queue 0		0	1	6
VMDBKs loading 0		0	0	0
Eligible VMDBKs 0		0	0	0
El. VMDBKs loading 0		0	0	0
Tot. WS (pages) 0		871	1653k	
Expansion factor 0		0	0	
85% elapsed time 4.712		.589	4.712	28.27
User Status:				
# of logged on users 25				
# of dialed users 0				
# of active users 11				
# of in-queue users 7				
% in-Q users in PGWAIT 0				
% in-Q users in IOWAIT 0				
% elig. (resource wait) 0				
Transactions Q-Disp trivial non-trv				
Average users .2		.1	.3	
Trans. per sec. .7		.6	.1	
Av. time (sec) .307		.258	3.061	
UP trans. time .258		3.061		
MP trans. time .000		.000		
System ITR (trans. per sec. tot. CPU) 1.4				
User Extremes:				
Max. CPU % LXOR4 95.1				
Max. VECT %				
Max. IO/sec LXOR4 113				
Max. PGS/s LXOR4 9.6				
Max. RESPG LXORINST 511413				
Max. MDCIO LXOR4 1.9				
Command ==>				
F1=Help F4=Top F5=Bot F7=Bkwd F8=Fwd F12=Return				

Figure 2-17 IBM z/VM Performance Toolkit screen 1, CPU load and trans. FCX100

Depending on the result we got from the first screen, we can now look into other areas of the IBM z/VM Performance Toolkit menu and gather information to use for our problem determination. For example, to get more information about the processor resource consumption of each user ID we display the 21- User resource usage page. This display is shown in Figure 2-18.

FCX112	CPU 2094		SER 7991E	Interval 11:26:27 - 11:27:27						Perf. Monitor		
	<----- CPU Load ----->				<----- Virtual IO/s ----->							
	<-Seconds->				T/V							
Userid	%CPU	TCPU	VCPU	Ratio	Total	DASD	Avoid	Diag98	UR	Pg/s	User	Status
>>Mean>>	3.87	2.321	2.298	1.0	7.7	7.7	.3	.0	.0	.1	---	---
DISKACNT	0	0	0	...	0	0	0	0	0	0	ESA,	---
DTCVSW1	.00	.000	.0000	.0	.0	.0	.0	.0	ESA,	---
DTCVSW2	.00	.000	.0000	.0	.0	.0	.0	.0	ESA,	---
EREP	0	0	0	...	0	0	0	0	0	0	ESA,	---
FTPSERVE	0	0	0	...	0	0	0	0	0	0	XC,	---
LNXINST	.03	.016	.015	1.1	.0	.0	.0	.0	.0	.0	EME,CL1,	DIS
LNXMAINT	0	0	0	...	0	0	0	0	0	0	ESA,	---
LXORINST	.40	.239	.222	1.1	.4	.4	.0	.0	.0	.0	EME,CL2,	DIS
LXOR1	.18	.107	.093	1.2	.0	.0	.0	.0	.0	.0	EME,CL3,	DIS
LXOR3	.18	.108	.100	1.1	.1	.1	.0	.0	.0	.0	EME,CL3,	DIS
LXOR4	94.1	56.43	56.01	1.0	181	181	1.0	.0	.0	2.6	EME,CL3,	DIS
MAINT	0	0	0	...	0	0	0	0	0	0	ESA,	---
OPERATOR	0	0	0	...	0	0	0	0	0	0	ESA,	---
OPERSYMP	0	0	0	...	0	0	0	0	0	0	ESA,	---
PERFSVM	.01	.005	.004	1.3	.2	.2	.1	.0	.0	.1	ESA,	---
TCPIP	.01	.003	.002	1.5	.0	.0	.0	.0	.0	.0	ESA,	---
TCPMAINT	0	0	0	...	0	0	0	0	0	0	ESA,	---
TEACHOR1	.69	.416	.379	1.1	9.5	9.5	5.8	.0	.0	.0	EME,CL3,	DIS
TEACHOR2	1.16	.694	.634	1.1	.5	.5	.0	.0	.0	.0	EME,CL3,	DIS
USER1	0	0	0	...	0	0	0	0	0	0	ESA,	---
USER10	0	0	0	...	0	0	0	0	0	0	ESA,	---
USER15	0	0	0	...	0	0	0	0	0	0	ESA,	---
VMSEVR	0	0	0	...	0	0	0	0	0	0	ESA,	---
Select a user for user details or IDLEUSER for a list of idle users												
Command ==>												
F1=Help F4=Top F5=Bot F7=Bkwd F8=Fwd F10=Left F11=Right F12=Return												

Figure 2-18 Performance Toolkit screen 21, User resource usage FCX112

If, for example, a Linux system is consuming a lot of CPU resources, we can drill deeper and display more detailed information about this specific user ID. To do so place the cursor on the user ID that you want to have more information about

and press Enter. In our example we want to get more information about user ID LXOR4. The screen that appears is as shown in Figure 2-19 on page 47.

FCX115	CPU 2094	SER 7991E	Interval 11:44:02 - 11:44:37	Perf. Monitor
Detailed data for user LXOR4				
Total CPU	: 93.5%	Storage def.	: 2200MB	Page fault rate: 1.8/s
Superv. CPU	: 1.7%	Resident <2GB:	105456	Page read rate : 16.5/s
Emulat. CPU	: 91.8%	Resident >2GB:	424406	Page write rate: .1/s
VF total	:%	Proj. WSET	: 544391	Pgs moved >2GB>: .0/s
VF overhead	:%	Reserved pgs	: 0	Main > XSTORE : 99.9/s
VF emulation	:%	Locked pages	: 6	XSTORE > main : 15.1/s
VF load rate	:s	XSTORE dedic.:	OMB	XSTORE > DASD : .1/s
I/O rate	: 410/s	XSTORE pages	: 16362	SPPOOL pg reads : .1/s
DASD IO rate	: 406/s	DASD slots	: 22663	SPPOOL pg writes: .0/s
UR I/O rate	: .0/s	IUCV X-fer/s	: .0/s	MDC insert rate: 344/s
Diag. X'98'	: .0/s	Share	: 100	MDC I/O avoided: 81.4/s
*BLOCKIO	: .0/s	Max. share	: ...	
#I/O active	: 0	Active	:100%	PSW wait : 0% I/O act. : 29%
Stacked blk	: ..	Page wait	: 7%	CF wait : 0% Eligible : 0%
Stat.: EME,DSC,RNBL		I/O wait	: 7%	Sim. wait: 0% Runnable : 86%
Data Space Name		Size Mode	PgRd/s PgWr/s XRd/s XWr/s Migr/s Steal/s	
BASE		2200MB Priv	16.5 .1 15.1 99.9 .1 101	
Device activity and status:				
0009 3215	.0		000C 254R	CL *, EOF NOH NCNT
000D 254P		CL A, CO 01, NOH NCNT	000E 1403	CL A, CO 01, NOH NCNT
0190	.0	CF31,RR, 107Cyl,--->0	0191	.0 CF37,WR, 10Cyl,--->0
019D	.0	CF34,RR, 146Cyl,--->0	019E	.0 CF34,RR, 250Cyl,--->0
0200 3390	.0	8025,WR,3338Cyl,RS/RL	0201	.7 D437,WR,10016Cy,RS/RL
0300	.4	D438,WR,10016Cy,RS/RL	0301	1.0 D439,WR,10016Cy,RS/RL
0302	.0	D43A,WR,10016Cy,RS/RL	0400	403 D43B,WR, 20% MDC eff.
0401	1.3	DB10,WR,3338Cyl,--->2	0402	.0 CF34,RR, 146Cyl,--->0
0405	.0	CF34,RR, 156Cyl,--->0	0500	.0 DA11,WR, 450Cyl,RS/RL
0501	.0	DA11,WR, 450Cyl,RS/RL	0502	.0 DA11,WR, 450Cyl,RS/RL
0503	.0	DA11,WR, 450Cyl,RS/RL	0504	.0 DA11,WR, 450Cyl,RS/RL
0600 OSA	.0	QDIO->TEACHOR1 QDIO	0601 OSA	.0 QDIO->TEACHOR1 QDIO
0602 OSA	3.5	QDIO->TEACHOR1 QDIO	0650 OSA	.0 QDIO->LXOR3 QDIO
0651 OSA	.0	QDIO->LXOR3 QDIO	0652 OSA	.0 QDIO->LXOR3 QDIO
0700	.0	D436,WR,10016Cy,RS/RL	0701	.0 DA14,WR,3338Cyl,RS/RL
0800	.0	DA12,WR,1500Cyl,RS/RL	0900	.0 DB12,RR,10016Cy,>4886
0901	.0	DB13,RR,10016Cy,--->0	0902	.0 DB14,RR,10016Cy,>1979
Command ==>				
F1=Help F4=Top F5=Bot F7=Bkwd F8=Fwd F12=Return				

Figure 2-19 Detailed data for user ID LXOR4, screen FCX115

This display can be refreshed any time by pressing the Enter key. In this screen, we see the detailed status of the user ID LXOR4. We can therefore identify possible problems like looping, high I/O rates, and so on. We can get more detailed information from this screen about DASD users. To do this, place the cursor in the appropriate field in the screen and press Enter. In our example, we want more information about the DASD with the CHPID 0201. The result is shown in Figure 2-20 on page 49.

FCX110 CPU 2094 SER 7991E Interval 16:31:00 - 16:31:02 Perf. Monitor

Detailed Analysis for Device D437 (SYSTEM)

Device type :	3390-9	Function pend.:	...ms	Device busy :	...%
VOLSER :	LXD437	Disconnected :	...ms	I/O contention:	...%
Nr. of LINKs:	2	Connected :	...ms	Reserved :	...%
Last SEEK :	8011	Service time :	...ms	SENSE SSCH :	...
SSCH rate/s :	.0	Response time :	...ms	Recovery SSCH :	...
Avoided/s :	.0	CU queue time :	...ms	Throttle del/s:	...

Status: MDCACHE USED

Path(s) to device D437:	8C	9C	8E	A4	84	9D	86	A5
Channel path status :	ON	ON	OFF	OFF	OFF	OFF	ON	ON

Device		Overall CU-Cache Performance							Split		
DIR	ADDR	VOLSER	IO/S	%READ	%RDHIT	%WRHIT	ICL/S	BYP/S	IO/S	%READ	%RDHIT
E4	D437	LXD437	.5	0	0	100	.0	.0	.3	0	0 (N)
									.1	0	0 (S)
									.0	0	0 (F)

MDISK Extent		Userid	Addr	IO/s	VSEEK	Status	LINK	VIO/s	%MDC
Nr. of LINKs:	2	Connected	:	...ms	Reserved	:	...		
Last SEEK :	8011	Service time :	SENSE SSCH :	...			
SSCH rate/s :	.0	Response time :	Recovery SSCH :	...			
Avoided/s :	.0	CU queue time :	Throttle del/s:	...			

Status: MDCACHE USED

Path(s) to device D437:	8C	9C	8E	A4	84	9D	86	A5
Channel path status :	ON	ON	OFF	OFF	OFF	OFF	ON	ON

Device		Overall CU-Cache Performance							Split		
DIR	ADDR	VOLSER	IO/S	%READ	%RDHIT	%WRHIT	ICL/S	BYP/S	IO/S	%READ	%RDHIT
E4	D437	LXD437	.5	0	0	100	.0	.0	.3	0	0 (N)
									.1	0	0 (S)
									.0	0	0 (F)

MDISK Extent	Userid	Addr	IO/s	VSEEK	Status	LINK	VIO/s	%MDC
+-----								
C	1 - 10016	LXORINST	021B	.0	0 WR	2	.0	...
+-----								

Command ==>

F1=Help F4=Top F5=Bot F7=Bkwd F8=Fwd F10=Left F11=Right F12=Return

Figure 2-20 Detailed information for DASD 0201 of user ID LXOD4

The REDISP screen allows you to view a maximum of the current day's history data simultaneously, but you have to detect any correlations between different variables yourself. To get into the REDISP screen type the command **redis** or simply press the PF2 key from the IBM z/VM Performance Toolkit main menu. The screen displayed looks similar to Figure 2-21 on page 51. Every 60 seconds a new set of variables is written. There are more columns of data available than displayed on the screen. To navigate to the left or right use the PF10 and PF11 keys. Note the ESCN field. This is a good indicator to see whether the system is in emergency scanning mode. The value can be between 0 and 100% (the percent of demand scans that did not complete with scan 1). We discuss a scenario where emergency scanning occurs in Chapter 11, "Case study: slow responding Web site" on page 313.

FCX101 CPU 2094 SER 7991E Interval 09:19:00 - 10:54:27 Perf. Monitor															
TIME >	PPAG	%ST	ALO/S	FPGS	%FR	SHAR	#TW	ESCN	%PGSL	%SPSL	XSTAV	%XS	XAL/S	XAG	
10:04	2265k	0	72	...	0	707k	0	53	100	52	2048M	100	168	56	
10:05	2265k	0	74	...	0	708k	0	..	100	52	2048M	99	2	103	
10:06	2265k	0	34	...	0	709k	0	..	100	52	2048M	100	92	185	
10:07	2265k	0	433	...	0	709k	1	..	98	52	2048M	99	0	330	
10:08	2265k	0	825	...	0	708k	2	99	81	52	2048M	96	374	394	
10:09	2265k	0	57	...	0	708k	0	94	81	52	2048M	97	150	224	
10:10	2265k	0	340	...	0	707k	4	80	85	52	2048M	100	935	107	
10:11	2265k	0	198	...	0	707k	7	78	89	52	2048M	100	252	124	
10:12	2264k	0	606	...	0	707k	9	84	92	52	2048M	100	266	165	
10:13	2265k	0	157	...	0	708k	7	90	94	52	2048M	100	222	182	
10:14	2265k	0	7	...	0	708k	0	100	94	52	2048M	100	36	301	
10:15	2265k	0	12	...	0	709k	0	89	95	52	2048M	100	67	413	
10:16	2265k	0	9	...	0	709k	0	100	80	46	2048M	93	27	637	
10:17	2265k	0	0	...	0	709k	0	..	80	46	2048M	93	0	11	
10:18	2265k	0	7	...	0	709k	0	92	80	46	2048M	93	44	11	
10:19	2265k	0	39	...	0	709k	1	89	80	46	2048M	95	173	654	
10:20	2265k	0	34	...	0	709k	1	69	80	46	2048M	96	157	366	
10:21	2265k	0	52	...	0	709k	6	22	80	46	2048M	98	189	359	
10:22	2265k	0	2	...	0	709k	0	..	80	46	2048M	97	0	590	
10:23	2265k	0	3	...	0	709k	0	..	80	46	2048M	97	0	11	
10:24	2265k	0	49	...	0	709k	0	..	80	46	2048M	96	0	18	
10:25	2265k	0	14	...	0	710k	0	0	80	46	2048M	96	5	30	
10:26	2265k	0	11	...	0	710k	0	33	80	46	2048M	96	19	27	
10:27	2265k	0	0	...	0	710k	0	..	80	46	2048M	96	0	33	
10:28	2264k	0	159	...	0	709k	1	41	82	46	2048M	100	430	265	
10:29	2264k	0	99	...	0	709k	0	53	84	46	2048M	100	141	324	
Enter 'GRAPHIcs' command for history graphics selection menu															
Command ==>															
F1=Help F4=Top F5=Bot F7=Bkwd F8=Fwd F10=Left F11=Right F12=Return															

Figure 2-21 Remote performance log display (FCX101)

Note: More information about how to use the IBM z/VM Performance Toolkit for problem determination can be found in *Linux on IBM eServer zSeries and S/390: Performance Toolkit for VM*, SG24-6059, and in *z/VM Performance Toolkit Guide Version 5 Release 3*, SC24-6156-01.

2.3 IBM Tivoli OMEGAMON XE on z/VM and Linux

Tivoli OMEGAMON® XE on z/VM and Linux is one of a suite of Tivoli Management Services products called Tivoli Enterprise Monitoring Agents. These products use common shared technology components to monitor your mainframe and distributed systems on a variety of platforms, and to provide workstation-based reports that you can use to track trends and understand and troubleshoot system problems.

Tivoli OMEGAMON XE on z/VM and Linux V4.1.0 is a product that gives you the ability to view data collected from multiple systems on a flexible interface. With this monitoring agent, you can view z/VM data obtained from the IBM z/VM Performance Toolkit (also known as the Performance Toolkit). You can also display Linux on System z performance data. This dual capability allows you to solve problems quickly and helps you to better and more easily manage a complex environment.

OMEGAMON monitoring software provides a portal that is accessible via a desktop client or a browser. This portal interface allows users to monitor systems without having to be logged onto either Linux or z/VM. The portal provides a single point of control for monitoring all of your Linux and z/VM images as well as z/OS® and its subsystems, databases, Web servers, network services, and WebSphere® MQ.

The OMEGAMON portal displays the following types of data:

- ▶ Memory usage, disk input and output (I/O) usage, and paging activity by workload name
- ▶ System-wide data by resource, such as logical partition usage and paging data
- ▶ System-wide spooling space usage
- ▶ TCP/IP network statistics for enabled network server virtual machines
- ▶ HiperSockets™ utilization data
- ▶ z/VM CPU data
- ▶ Linux on IBM System z workload statistics

The beginning phase of checking the performance of a system, or multiple systems, using the OMEGAMON product is to look at the high-level displays for each system. In our environment we are only concerned with the performance of z/VM and its Linux guests. Since z/VM controls all of the hardware resources and virtualizes that environment for the Linux guests, the OMEGAMON display for z/VM is a good starting point. The following series of graphs from the

OMEGAMON display highlight the key performance indicators of CPU, disk, and storage utilizations.

Figure 2-22 demonstrates a graph of CPU utilization displayed by OMEGAMON.

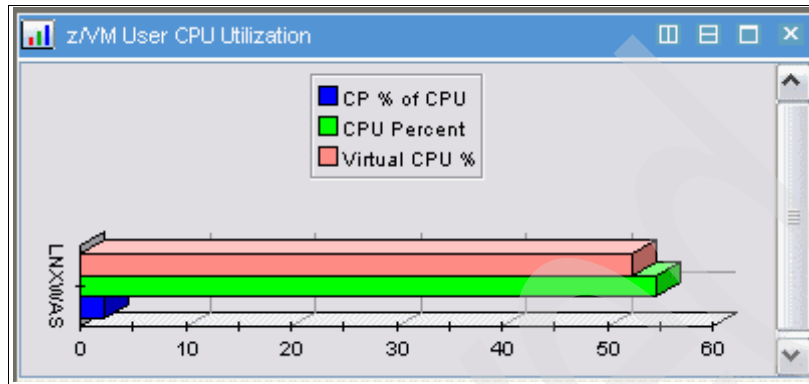


Figure 2-22 OMEGAMON CPU utilization graph

Figure 2-23 demonstrates a graph of device utilization displayed by OMEGAMON.

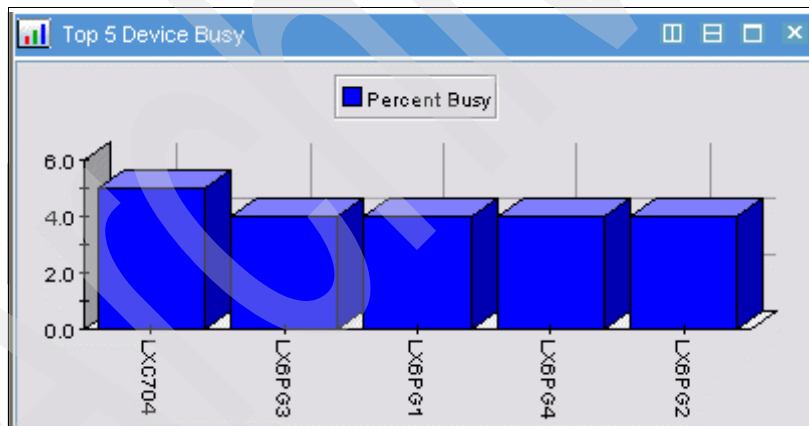


Figure 2-23 OMEGAMON device utilization graph

Figure 2-24 demonstrates a graph of storage utilization displayed by OMEGAMON.

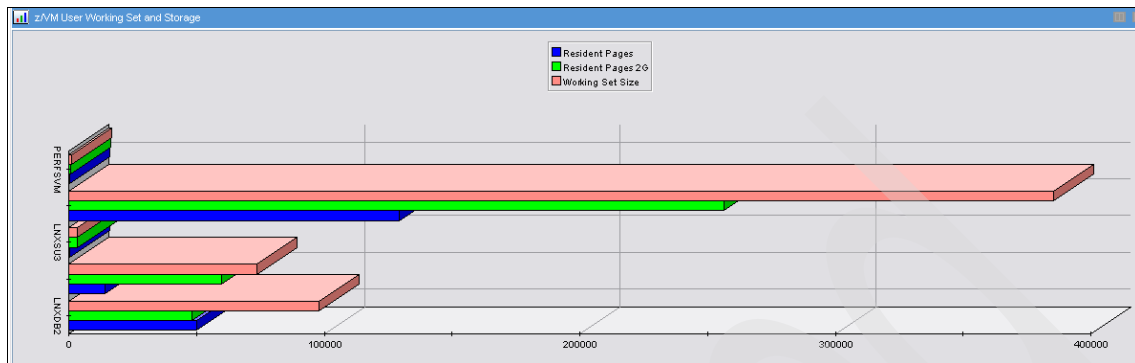


Figure 2-24 OMEGAMON storage utilization graph

The same data can be displayed from the OMEGAMON portal in text or tabular form through options on the panels. In a System z environment, it is possible to share processors between LPARS, which means that the processors assigned to this z/VM system might be shared by other z/VM, Linux, or z/OS systems. This means that an investigation of system performance must include a view of the total hardware complex. The graph shown in Figure 2-25 from OMEGAMON shows the data from our total hardware system.

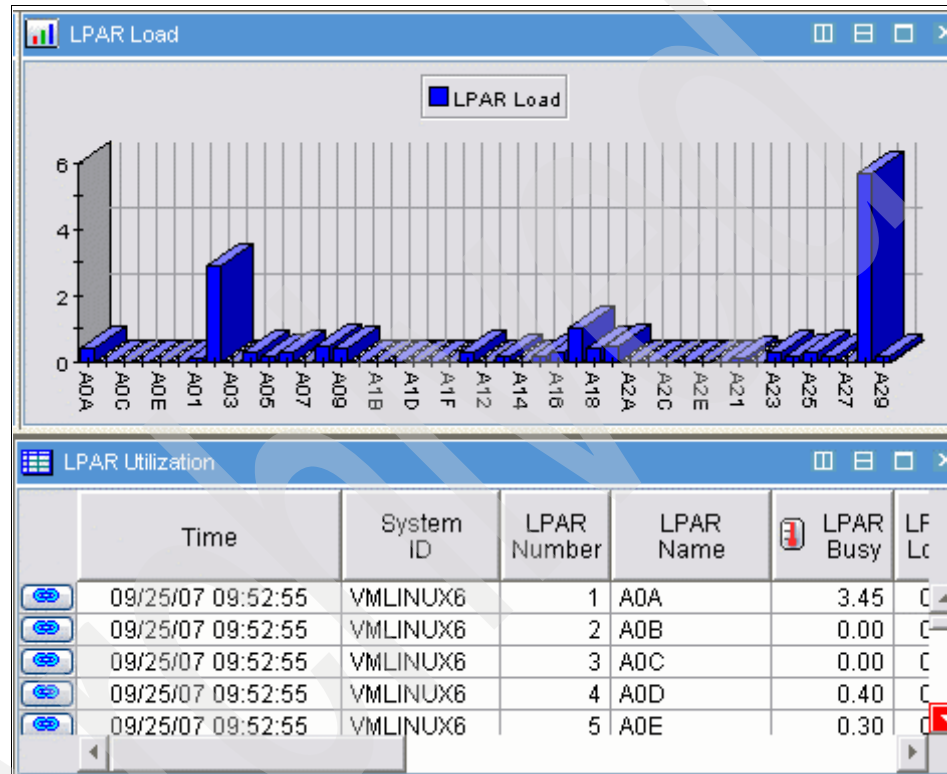


Figure 2-25 OMEGAMON LPAR utilization graph

The LPAR utilization shown in Figure 2-25 has a bar chart showing processor utilization of every LPAR on the System z, and it has a table with detailed information about each LPAR. The table has scroll bars to expand the table and show more columns and rows. On the left of each row is a link icon, which connects a user to additional information relating to that row (or LPAR).

After careful inspection of the system performance information available via the z/VM displays from OMEGAMON, the next step in a performance evaluation is to look at statistics from inside one or many of the Linux guests that are running on z/VM.

The overview workspace for Linux guests displays many of the key performance indicators. In Figure 2-26, OMEGAMON shows the top Linux systems and their CPU utilization, their load on the system for the last 15 minutes, and a table of system statistics (context switches, paging rates, length of time that the system has been up, and number of users logged on).

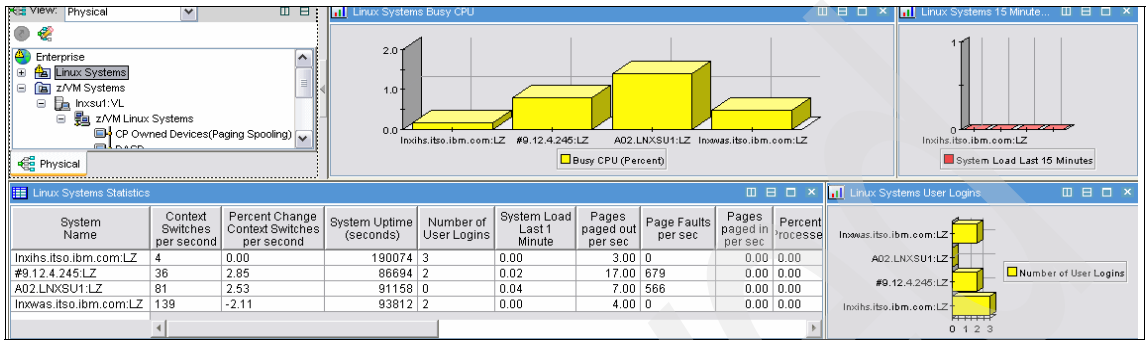


Figure 2-26 OMEGAMON Linux guest performance overall view

After reviewing the high-level information that is available on the collective Linux display, additional OMEGAMON panels show details on Linux systems, processes, files, and networks. Some of those panels are shown in Figure 2-27 and Figure 2-28 on page 57.

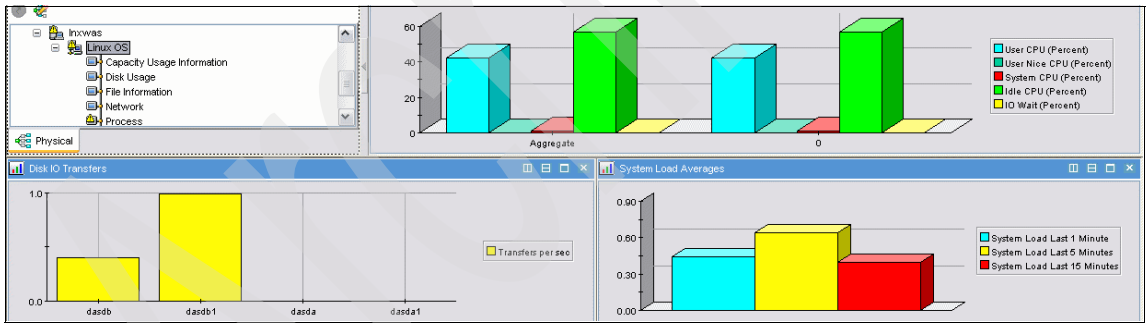


Figure 2-27 OMEGAMON Linux system graph

Linux Guest Appl Data																
Time	System ID	LPAR Name	User ID	Virtual CPUs	Total CPU	User CPU	Kernel CPU	Nice CPU	Percent IRQ	Percent Soft IRQs	Percent I/O Wait	Percent CPU Idle	Runnable Processes	Processes Waiting for I/O	Total Processes	Avg P Las
09/25/07 11:25:56	VMLINUX6	A02	LNxDB2	1	15.20	9.00	5.60	0.00	0.30	0.30	2.80	81.30	5	1	194	
09/25/07 11:25:56	VMLINUX6	A02	LNXIHS	1	1.70	0.90	0.40	0.00	0.20	0.30	0.10	98.20	2	0	160	
09/25/07 11:25:56	VMLINUX6	A02	LNXSU1	1	2.20	1.79	0.40	0.00	0.00	0.00	0.00	97.80	2	0	286	
09/25/07 11:25:56	VMLINUX6	A02	LNKWAS	1	5.30	5.00	0.20	0.00	0.00	0.00	0.50	94.00	30	0	202	

Figure 2-28 OMEGAMON Linux application table

Figure 2-29 shows the status and resource utilization for each process. Tables such as this in OMEGAMON can be filtered (to eliminate meaningless data) and sorted (to place data of high interest at the top).

Process Information Detail													
	Process Command name (Unicode)	Process ID	Process Parent ID	Process State	Process System CPU (Percent)	Process User CPU (Percent)	Cumulative Process System CPU (Percent)	Cumulative Process User CPU (Percent)	Kernel Priority	Nice Value	Total Size(pages)	Resident Set Size(pages)	M
	cupsd	1244	1	Sleeping	594.07	0.05	0.00	0.00	16	0	2300	1091	
	portmap	1231	1	Sleeping	594.07	0.00	0.00	0.00	15	0	462	122	
	cron	1240	1	Sleeping	594.07	0.00	0.00	0.00	16	0	558	212	
	gengat	1206	1	Sleeping	969.02	0.02	0.00	0.00	16	0	871	251	

Figure 2-29 OMEGAMON Linux processes table

With the Tivoli OMEGAMON XE for z/VM and Linux agent, you can also perform the following tasks:

- ▶ Navigate to greater levels of detailed performance data. For Linux guests, this monitoring agent provides links to Tivoli Monitoring Agent for Linux OS workspaces directly from the Tivoli OMEGAMON XE on z/VM and Linux workspaces.
- ▶ Connect to the z/VM host system by means of TCP/IP to access the Performance Toolkit panels.
- ▶ View expert advice on how to identify and resolve performance problems.

IBM Tivoli OMEGAMON comes with default thresholds for key system performance indicators. When a threshold is exceeded, a situation alert is generated and an automated action is taken to address the problem. The situation console log workspace shows the active situation alerts. In Figure 2-30 the upper panel lists open situations. The lower left panel has the situation count for the last 24 hours. The lower right panel lists the situation events that have been acknowledged by the current user.

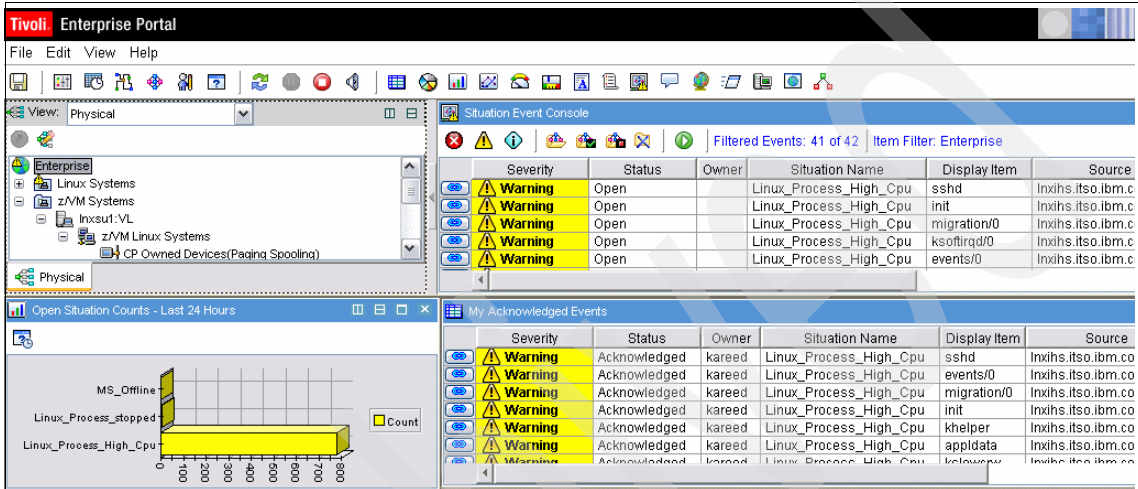


Figure 2-30 OMEGAMON situation alerts

Default workspaces and situations enable you to start monitoring your enterprise as soon as the Tivoli OMEGAMON XE on z/VM and Linux software is installed and configured. The user interface supports several formats for viewing data, such as graphs, bar charts, and tables. Workspaces and situations can be customized to meet the needs of your enterprise.

Problem determination tools for Linux on System z

In this chapter we discuss tools that are used during problem determination on Linux on System z. These tools are available on both Linux and z/VM for problem determination. The problem area or category will decide which tools will be used to help determine the problem. We always recommend using z/VM-based tools or a combination of z/VM and Linux-based tools to get more granular information about the problem.

The Linux environment has several specialized tools for the purpose of problem determination. In this chapter we explore the information gathered and the circumstances in which the tools would be useful for problem determination.

3.1 Information-gathering tools

As discussed in the previous chapter, the first and foremost thing that we need to do before starting to use any of the specialized debugging tools is to investigate the problem cause. For that we need to gather information about the environment, resources, and other configurations. In this section we discuss various Linux-based tools that are available.

Note: The important thing here with the Linux-based tools is that the resource utilization reported by any of the Linux-based tools has to be correctly understood, keeping in mind the underlying z/VM resource management. When Linux runs as a guest, z/VM time shares the Linux images and also optimally shares the resource pool. Linux tools only see the share that they got in z/VM. We highly recommend that resource utilization information gathered on Linux-based tools be cross-checked with the underlying z/VM-based tools.

Usually, most of these tools provide utilization information for a certain period of time, but there are also tools that only provide snapshot utilization. So in a problem determination approach, we recommend using shell scripts to record the utilization and time stamp for a certain period, for the tools that can only report snapshot data. By doing this, you will be able to understand the transition of utilization for a certain period. Also, the timestamps will help you to compare the information gathered on various tools.

3.1.1 dbginfo.sh

For collecting the overall snapshot information about the Linux on System z environment, you are provided with the dbginfo.sh script, which is part of the s390-tools rpm package, which also comes as a default package with the SUSE® and RHEL Linux distributions. Some of the information gathered by this script is:

- ▶ Information from the proc file system
[version,cpu,meminfo,slabinfo,modules,partitions,devices ...]
- ▶ System z specific device driver information: /proc/s390dbf
- ▶ Kernel messages /var/log/messages
- ▶ Configuration files /etc/[ccwgroup.conf,chandev.conf,modules.conf,fstab]
- ▶ Several commands: **ps**, **vgdisplay**, **dmesg**

Example 3-1 Example dbginfo.sh execution list

```
lnx02:/ # dbginfo.sh
Create target directory /tmp/DBGINFO-2008-03-13-19-14-59-lnxihs
Change to target directory /tmp/DBGINFO-2008-03-13-19-14-59-lnxihs
Get procfs entries
Saving runtime information into runtime.out
Get file list of /sys
Get entries of /sys
Check if we run under z/VM
  Running under z/VM
  Installed CP tool: vmcp
Saving z/VM runtime information into zvm_runtime.out
Copy config files
Copy log files

Collected data was saved to:
  /tmp/DBGINFO-2008-03-13-19-14-59-lnxihs.tgz
```

The dbginfo.sh script is also available for download from the IBM developerWorks® Web site:

<http://www.ibm.com/developerworks/linux/linux390/s390-tools-1.6.1.html>

3.1.2 vmstat

The **vmstat** command displays current statistics for processes, usage of real and virtual memory, paging, block I/O, and CPU. The left-most columns show the number of running processes and number of blocked processes. The memory section shows memory being swapped out, free memory, the amount of buffer containing inodes and file metadata, and cached memory for files being read from disk. The swap section lists swaps in and swaps out. The I/O section reports the number (kb) of blocks read in and written out. System in and cs represent interrupts and context switches per second. The CPU section headers of us, sy, id, and wa represent the percentage of CPU time count on users, system, idle, I/O wait, and steal, respectively. The **vmstat** command is useful in identifying memory shortages and I/O wait issues, in addition to situations where virtual CPUs are not backed up by physical CPUs. Also, **vmstat** has a low performance overhead during the information-gathering phase.

In Example 3-2 we see that there is heavy paging taking place. In our case even the swap space is fully used up. It might be the cause of the degradation in the application transaction rate and response time, in which case Linux might be spending its CPU cycles on swapping in and out the pages. In this case we did a over commitment of memory to the affected Linux instance to improve the response time.

Example 3-2 Gathering swap, memory, and CPU information using vmstat

```
lnx02:~ # vmstat 30
```

procs		-----memory-----				---swap--		-----io-----		-system--		-----cpu-----				
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
0	0	713640	7784	1932	28644	0	0	0	7	152	142	0	0	99	0	0
0	0	713632	7784	1972	29096	1	0	1	5	158	141	0	0	99	0	0
0	0	713616	7536	2012	29040	1	0	1	8	159	143	0	0	99	0	0
0	0	713600	7536	2052	28984	1	0	1	3	161	140	0	0	99	0	0
0	0	713596	7412	2092	28916	1	0	1	8	156	143	0	0	99	0	0
0	0	713596	7412	2132	28876	0	0	0	2	152	140	0	0	99	0	0
0	0	719888	9144	2176	29316	1	210	1	217	257	142	0	0	98	2	0
0	0	718812	12976	2212	29316	122	26	122	32	186	174	1	0	97	1	0
0	0	718668	7340	2240	29228	360	164	360	172	270	259	1	1	93	5	0
1	0	719172	8764	2044	26808	789	453	789	456	360	339	2	2	86	10	0
0	0	718160	10364	808	21968	382	213	416	221	349	346	3	2	91	5	0
0	0	719400	10164	824	21124	264	200	264	202	425	366	3	1	92	4	0

3.1.3 System status (sysstat) tool

This package consists of several Linux tools to collect system data. The **sysstat** package is a widely used Linux standard tool. It is included in your distribution or can be downloaded from the Web:

<http://pagesperso-orange.fr/sebastien.godard/>

If you install the source package, you must compile the package on Linux on System z to use it. The sysstat package consists of the following components:

- ▶ **sar** reporting tool - reads binary file created with 'sadc' and converts it to readable output
- ▶ **sadc** data gatherer - stores data in binary file
- ▶ **iostat** - I/O utilization
- ▶ **mpstat** - processor utilization

For a more detailed description go to:

http://www.ibm.com/developerworks/linux/linux390/perf/tuning_how_tools.html#sysstat

sar

The **sar** command collects and displays system data. It writes to standard output the contents of selected cumulative activity counters in the operating system. The accounting system, based on the values in the count and interval parameters, writes information at the specified number of times spaced at the specified intervals in seconds.

With its numerous options, the **sar** command provides queuing, paging, TTY, and many other statistics. One important feature of the **sar** command is that it reports either system-wide (global among all processors) CPU statistics (which are calculated as averages for values expressed as percentages, and as sums otherwise) or it reports statistics for each individual processor.

The **sar** command prints the following information:

- ▶ Process creation
- ▶ Context switching
- ▶ All/single CPU utilization
- ▶ Network utilization
- ▶ Disk statistics

In Example 3-3 the **sar** command with the **-d** option generates real-time disk I/O statistics.

Example 3-3 Gathering disk information using sar

```
lnx02:~ # sar -d 3 3
Linux 2.6.16.46-0.12-default (lnxwas) 03/14/08
18:55:55      DEV      tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz   await  svctm   %util
18:55:58  dev94-0  0.00      0.00      0.00      0.00      0.00      0.00  0.00   0.00
18:55:58  dev94-4  0.00      0.00      0.00      0.00      0.00      0.00  0.00  0.00   0.00

18:55:58      DEV      tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz   await  svctm   %util
18:56:01  dev94-0  0.00      0.00      0.00      0.00      0.00      0.00  0.00  0.00   0.00
18:56:01  dev94-4  0.66      0.00     26.58     40.00      0.00      5.00  5.00   0.33

18:56:01      DEV      tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz   await  svctm   %util
18:56:04  dev94-0  0.00      0.00      0.00      0.00      0.00      0.00  0.00  0.00   0.00
18:56:04  dev94-4  0.00      0.00      0.00      0.00      0.00      0.00  0.00  0.00   0.00
```

Average:	DEV	tps	rd_sec/s	wr_sec/s	avgrq-sz	avgqu-sz	await	svctm	%util
Average:	dev94-0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Average:	dev94-4	0.22	0.00	8.85	40.00	0.00	5.00	5.00	0.11

sadc

The **sadc** command samples system data a specified number of times (count) at a specified interval measured in seconds (interval). It writes in binary format to the specified outfile or to the standard output.¹ The **sadc** command is intended to be used as a back end to the **sar** command.

Example 3-4 shows the syntax and an example of using the **sadc** command.

Example 3-4 Syntax and example of using sadc

```
/usr/lib64/sa/sadc <interval> <count> > out
/usr/lib64/sa/sadc 1 5 >out
```

Without the parameter *count*, **sadc** samples data until it is stopped. The standard output file is `/var/log/sa/sa<day>`, and it is written to once a day, at the end of the day.

iostat

In order to see an overview of your CPU utilization you can use the **iostat** command. This command provides monitoring and reporting on both CPU statistics and I/O statistics for one or more devices and partitions.

The following command analyzes I/O-related data for all disks:

```
iostat -dkx
```

Example 3-5 shows a sample of output from this command.

Example 3-5 Sample output from iostat command

Linux 2.6.9-67.sw (t6315015)		06/04/2008											
Device:	rrqm/s	wrqm/s	r/s	w/s	rsec/s	wsec/s	rkB/s	wkB/s	avgrq-sz	avgqu-sz	await	svctm	%util
dasda	3.41	14.09	20.35	11.20	1094.87	202.32	547.44	101.16	41.12	0.16	5.06	3.76	11.87

mpstat

In order to determine processor utilization you can use the **mpstat** command. The **mpstat** command writes to standard output the activities for each available processor, with processor 0 being the first one reported. Global average

¹ http://linuxcommand.org/man_pages/sadc8.html

activities among all of the processors are also reported.² **mpstat** displays more in-depth CPU statistics than **iostat**.

Automatic sampling via the cron

The **sadc** command collects system utilization data and writes it to a file for later analysis. By default, the data is written to files in the `/var/log/sa/` directory. The files are named `sa<dd>`, where `<dd>` is the current day's two-digit date.

The command **sadc** is normally run by the `sa1` script. This script is periodically invoked by cron via the file `sysstat`, which is located in `/etc/cron.d`. The `sa1` script invokes **sadc** for a single one-second measuring interval. By default, cron runs `sa1` every 10 minutes, adding the data collected during each interval to the current `/var/log/sa/sa<dd>` file.

The **sar** command produces system utilization reports based on the data collected by **sadc**. As configured in Red Hat Linux, **sar** is automatically run to process the files automatically collected by **sadc**. The report files are written to `/var/log/sa/` and are named `sar<dd>`, where `<dd>` is the two-digit representation of the previous day's two-digit date. The command **sar** is normally run by the `sa2` script. This script is periodically invoked by the cron via the file `sysstat`, which is located in `/etc/cron.d`. By default, the cron runs `sa2` once a day at 23:53, allowing it to produce a report for the entire day's data.

To activate the cron daemon, run the following command:

```
service crond start
```

To start the service for the indicated level:

```
chkconfig --level 123456 crond on
```

To check the status:

```
service --status-all |grep cron
chkconfig --list |grep cron
```

As soon as the cron daemon and the service are started, they use the definitions in `/etc/cron.d/sysstat`, as shown in Example 3-6.

Example 3-6 Automatic sampling via the cron

```
# run system activity accounting tool every 10 minutes
*/10 * * * * root /usr/lib64/sa/sa1 1 1
# generate a daily summary of process accounting at 23:53
53 23 * * * root /usr/lib64/sa/sa2 -A
```

² http://linuxcommand.org/man_pages/mpstat1.html

3.1.4 top

The **top** command displays process statistics and a list of the top CPU-using Linux processes. Options are available to interactively change the format and content of the display. The process list may be sorted by run time, memory usage, and processor usage. Top shows the total number of processes, the number running, and the number sleeping. Information is updated every three seconds, or at a chosen interval. This is useful in obtaining a snapshot of the processes that are running on a Linux system and their CPU and memory consumption.

Starting with SLES10 and RHEL5, the **top** command includes a new field, CPU steal time—the time Linux wanted to run, but z/VM gave the CPU to some other guest. This field will be very useful for understanding CPU performance characteristics from within Linux. Also, it is worth noting that **top** is a resource-consuming tool, which means that the performance overhead is comparably high when its is activated.

Example 3-7 Gathering process information using top

```
top - 20:59:57 up 8 min,  1 user,  load average: 0.68, 0.62, 0.25
Tasks:  53 total,   2 running, 51 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.3%us,  0.0%sy,  0.0%ni, 98.7%id,  0.7%wa,  0.3%hi,  0.0%si,  0.0%st
Mem:   3080964k total, 1029608k used, 2051356k free,   9780k buffers
Swap:   719896k total,    0k used,  719896k free,  234804k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1526	root	17	0	2437m	771m	56m	S	0.3	25.7	1:01.09	java
1	root	16	0	848	316	264	S	0.0	0.0	0:00.50	init
2	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
4	root	10	-5	0	0	0	S	0.0	0.0	0:00.03	events/0

In Example 3-7, we are able to easily ascertain that around 25% of the memory and some CPU cycles are being consumed by a Java™ process compared to the other processes running on the system.

3.1.5 ps

The **ps** command is a tool for identifying the programs that are running on the system and the resources that they are using. It displays statistics and status information about processes on the system, such as process or thread ID, I/O activity, CPU, and memory utilization. Also, we would be able to narrow down to the processes that dominate the usage.

Example 3-8 Gathering process information using ps

lnx02:~ #	ps au									
USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1464	0.0	0.0	2000	652	ttyS0	Ss+	16:08	0:00	/sbin/mingetty
--noclear /dev/ttyS0 dumb										
root	1468	0.0	0.0	5132	2652	pts/0	Ss	16:08	0:00	-bash
root	2007	8.7	26.0	2507240	801100	pts/0	Sl	16:40	1:07	/opt/java/bin/java
root	2190	0.0	0.0	2680	988	pts/0	R+	16:53	0:00	ps au

ps is a handy command that can be used with various combinations of parameters to obtain relevant information required on the processes. For more information about the various parameters, refer to the respective **ps** man pages.

3.1.6 ipcs

The **ipcs** command reports information about active interprocess communication facilities. The report will have information about currently active message queues, shared memory segments, and semaphores. The **ipcs** command is a very powerful tool that provides a look into the kernel's storage mechanisms for IPC objects.

Example 3-9 Gathering Inter process communication (IPC) information using IPCS

```
lnx02:~ # ipcs
----- Shared Memory Segments -----
key          shmid      owner      perms      bytes      nattch     status
0x0105e8ae  98304      root       660        52428800   1

----- Semaphore Arrays -----
key          semid      owner      perms      nsems
0x0105e8ad   0         root       660        3

----- Message Queues -----
key          msqid      owner      perms      used-bytes   messages
```

3.1.7 iostat

The **iostat** command provides CPU and input/output statistics for the devices and partitions. Primarily, the **iostat** command is used for monitoring system input/output device by observing the time that the devices are active in relation to their average transfer rates.

Begin the assessment by running the **iostat** command with an interval parameter during your system's peak workload period or while running a critical application for which you need to minimize I/O delays.

Example 3-10 Gathering input/output information using iostat

```
lnxwas:~ # iostat 5 3
Linux 2.6.16.46-0.12-default (lnxwas) 03/17/08
```

avg-cpu:	%user	%nice	%system	%iowait	%steal	%idle
	2.34	0.18	0.31	1.21	0.12	95.84
Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn	
dasdb	4.62	91.15	21.18	708616	164648	
dasda	0.01	0.08	0.00	632	0	
avg-cpu:	%user	%nice	%system	%iowait	%steal	%idle
	56.69	0.00	2.99	0.40	2.79	36.93
Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn	
dasdb	0.20	3.19	0.00	16	0	
dasda	0.00	0.00	0.00	0	0	

In Example 3-10, there are lots of blocks being read and written to the DASDs. The statistics provided above are on a per-disk basis.

3.1.8 netstat

netstat is a powerful tool for checking your network configuration and activity. The **netstat** command displays information regarding traffic on the configured network interfaces, such as the following:

- ▶ The address of any protocol control blocks associated with the sockets and the state of all sockets
- ▶ The number of packets received, transmitted, and dropped in the communications subsystem
- ▶ Cumulative statistics per interface

► Routes and their status

Example 3-11 Gathering network traffic information using netstat

```
lnx02:~ # netstat -i
Kernel Interface table
Iface  MTU Met  RX-OK RX-ERR RX-DRP RX-OVR   TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0   1492  0 1067003    0    0    0 1053600    0    0    0 BMRU
lo     16436  0   355      0    0    0   355      0    0    0 LRU
```

When invoked with the `-i` flag, `netstat` displays statistics for the network interfaces currently configured. The MTU and Met fields show the current MTU and metric values for that interface. The RX and TX columns show how many packets have been received or transmitted error-free (RX-OK/TX-OK) or damaged (RX-ERR/TX-ERR). When `netstat` is executed with the `-ta` flag, it reports the active and passive connections on the system.

Example 3-12 Gathering network connection information using netstat

```
lnx02:~ # netstat -ta
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0 *:can-ferret            *:.*                    LISTEN
tcp      0      0 *:nfs                    *:.*                    LISTEN
tcp      0      0 *:51202                  *:.*                    LISTEN
tcp      0      0 *:8803                   *:.*                    LISTEN
tcp      0      0 *:filenet-cm             *:.*                    LISTEN
tcp      0      0 *:can-ferret-ssl         *:.*                    LISTEN
tcp      0      0 *:52237                  *:.*                    LISTEN
tcp      0      0 *:sunrpc                  *:.*                    LISTEN
tcp      0      0 *:ftp                     *:.*                    LISTEN
tcp      0      0 lnx02.itso.ibm.c:54630  lnx03.itso.ibm.c:55000 ESTABLISHED
```

3.1.9 DASD statistics

For collecting performance statistics on DASD activities, in Linux we have a DASD statistics option. It records DASD I/O activities for a specific period of time as statistical data. When the statistic option is enabled, the DASD driver collects the statistical data.

DASD statistics can be easily enabled or disabled by switching on/off in the proc file system. See Example 3-13.

Example 3-13 Enabling and disabling DASD statistics

```
echo set on > /proc/dasd/statistics
echo set off > /proc/dasd/statistics
```

In Example 3-14 we are reading the entire system's DASD statistics. By turning off and back on the DASD statistic option we can reset all counters.

Example 3-14 Gathering DASD I/O statistics using DASD statistic option

```
lnx02:~ # cat /proc/dasd/statistics
613 dasd I/O requests
with 9432 sectors(512B each)
  <4   8   16   32   64  128  256  512  1k   2k   4k   8k  16k  32k  64k 128k
  256  512  1M   2M   4M   8M  16M  32M  64M 128M 256M 512M 1G   2G   4G  >4G
Histogram of sizes (512B secs)
  0   0  472  107   6   18   9   1   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
Histogram of I/O times (microseconds)
  0   0   0   0   0   0   0   0  117  124  142  156  47  26   1   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
Histogram of I/O times per sector
  0   0   0   0  11  164  122  161  123  23   9   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
Histogram of I/O time till ssch
 145   50   13   0   0   0   0   0  74   67  238   3  16   7   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
Histogram of I/O time between ssch and irq
  0   0   0   0   0   0   0   0  454  98   7   1  39  13   1   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
Histogram of I/O time between ssch and irq per sector
  0   0   0   1  74  462  42  12   7  13   2   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
Histogram of I/O time between irq and end
 151  216  223  22   0   1   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
# of req in chang at enqueueing (1..32)
  0  207   86   64   51  204   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

Explanation of the output

Line 1 contains the total number of I/O requests (Start Subchannel (SSCH) instructions) processed during the observation period. In this example it is 613.

Line 2 contains the number of 512-byte blocks transferred during the observation period. In this example it is 9432.

Lines 3 and 4 contain either times (microseconds) or sizes (512-byte blocks), depending on the context.

See the following Web site for a more detailed explanation:

http://www.ibm.com/developerworks/linux/linux390/perf/tuning_how_tools_dasd.html

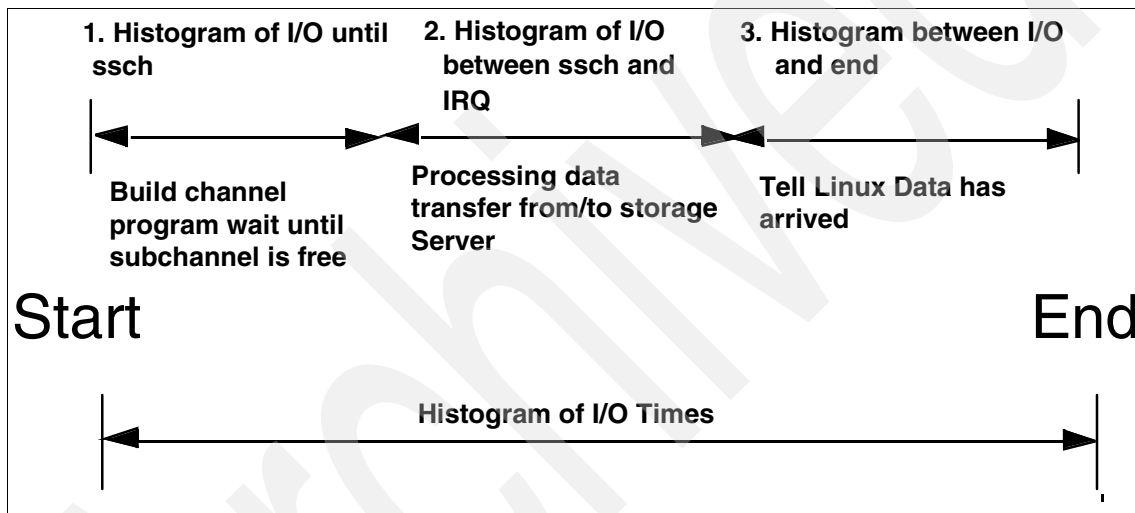


Figure 3-1 Explanation for the DASD statistics Histogram details

Example 3-15 on page 72 demonstrates the distribution of the total time needed for the I/O requests, starting at the point where the DASD driver accepts the request from the kernel until the DASD driver informs the kernel of the completion of the request. Each step along the way is measured as shown in Example 3-15 on page 72.

- ▶ The distribution of the total time needed for the I/O requests, starting at the point where the DASD driver accepts the request from the kernel until the DASD driver informs the kernel about the completion of the request.
- ▶ The distribution of the time that the I/O request is queued by the DASD driver. This time starts when the DASD driver accepts the I/O request from the kernel and ends when the SSCH is issued. Typically, these requests have to wait because the particular subchannel is busy.

- ▶ The distribution of the time between SSCH and channel end/device end (CE/DE), that is, the time needed by the channel subsystem and the storage subsystem to process the request. It helps to understand the performance of the storage subsystem.
- ▶ The distribution of the time the DASD driver needs from accepting CE/DE until presenting the IRQ to the kernel.

As we have discussed, the example above is for the overall system DASD statistics. For getting I/O statistical information about a particular DASD, we can use the **tunedasd** tool.

In Example 3-15, we issue the **tunedasd** command against our DASD where the root file system is mounted (that is, /dev/dasdb1).

Example 3-15 Gathering I/O information about specific DASD using tunedasd

```
lnxw02:/usr/src # tunedasd -P /dev/dasdb1
137609 dasd I/O requests
with 2419848 sectors(512B each)
__<4 __8 __16 __32 __64 __128 __256 __512 __1k __2k __4k __8k __16k __32k __64k 128k
__256 __512 __1M __2M __4M __8M __16M __32M __64M 128M 256M 512M __1G __2G __4G __>4G
Histogram of sizes (512B secs)
  0  0  6704 2223 3078 3177 1604 458 160 198  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Histogram of I/O times (microseconds)
  0  0  1204 2352 4727 1116 144  38 1419 1605 2350 2434 1003 575 139 52
142 53  1  2  0  0  0  0  0  0  0  0  0  0  0  0  0
Histogram of I/O times per sector
3777 3318  490 148 1258 2330 1586 2463 2025 670 317 51 13 2 1 1
 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Histogram of I/O time till ssch
6669 643 272 54 47 12 1 3 859 912 3469 155 312 225 78 31
87 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Histogram of I/O time between ssch and irq
  0  0 1699 2144 336 47 7 2 6260 1476 422 111 893 290 50 15
21 1 0 2 0 0 0 0 0 0 0 0 0 0 0 0
Histogram of I/O time between ssch and irq per sector
4026 194 18 30 1130 6525 656 365 205 501 98 6 3 2 0 1
 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Histogram of I/O time between irq and end
6735 3105 3541 257 86 22 8 1 3 4 5 6 1 0 0 0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
# of req in chang at enqueueing (1..32)
  0 7584 1154 957 752 3312 0 0 0 0 0 0 0 0 0 0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

3.1.10 OProfile

OProfile is an open source profiler for Linux systems. It offers profiling of all running code on a Linux system, including the kernel, shared libraries, application binaries, and so on, and provides a variety of statistics at a low overhead (varying from 1–8%) depending on the workload. It is released under the GNU GPL. OProfile consists of a kernel driver, a daemon for collecting sample data, and tools for report generation.

OProfile is generally supported in Linux kernel 2.2, 2.4, 2.6, and later. But, for Linux on System z, OProfile only supports kernel 2.6 or later. For SUSE Linux, the kernel level must be at least kernel-s390(x)-2.6.5-7.191, which starts from SLES9 SP2. For Red Hat Linux, the kernel level must be at least kernel-2.6.9-22.EL, which is RHEL4 U2.

OProfile utilizes CPU's performance counters to count events for all of the running code, and aggregates the information into profiles for each binary image. However, System z hardware currently does not have support for this kind of hardware performance counters utilized by OProfile, so the timer interrupt is used instead.

Using OProfile the code can be profiled for:

- ▶ Hardware and software interrupt handlers
- ▶ Kernel modules, the kernel
- ▶ Shared libraries
- ▶ Applications

OProfile setup involves configuring the kernel source by enabling the profiling option. To build and install OProfile, issue the commands shown in Example 3-16 from the decompressed oprofile directory. As shown in this example, we used the option *with-kernel-support*. You would use this option when you have a kernel Version of 2.6 and later. For more information about installation instructions of OProfile, see:

<http://oprofile.sourceforge.net/doc/install.html>

Example 3-16 Setting up OProfile

```
./configure --with-kernel-support
make
make install
```

To use the OProfile tool, the timer should be turned on by using command **sysctl**. After all the profiling is done, turn the timer off. Once OProfile is installed successfully, you must tell OProfile the location of the vmlinux file corresponding to the running kernel. If the kernel is not intended to be profiled then OProfile is set by passing a parameter telling the system that it does not have a vmlinux file. Both examples are shown in Example 3-17.

Example 3-17 Basic profiling configurations

```
opcontrol --vmlinux=/boot/vmlinux-`uname -r`  
or  
opcontrol --no-vmlinux
```

Once OProfile is installed and set up, you can use the **Opcontrol** command to start and stop (shut down) the profiling daemon (see Example 3-18).

Example 3-18 OProfile start and stop commands

```
opcontrol --start / --shutdown
```

Opcontrol takes a specification that indicates how the details of each hardware performance counter should be set up. Kernel or user space code is profiled based on these settings.

Example 3-19 shows a sample OProfile report.

Example 3-19 Sample OProfile profile report

```
lnx02:~ # opreport --symbols  
CPU: CPU with timer interrupt, speed 0 MHz (estimated)  
Profiling through timer interrupt  
samples %      image name      app name      symbol name  
10181    65.6288  lib_app1      lib_app1     main  
3390      21.8526  vmlinux        vmlinux       .text  
1694      10.9199  no-vmlinux     no-vmlinux    (no symbols)  
40         0.2578  vmlinux        vmlinux       cpu_idle  
39         0.2514  libc-2.4.so    libc-2.4.so   _int_malloc  
23         0.1483  vmlinux        vmlinux       get_page_from_freelist  
8          0.0516  vmlinux        vmlinux       do_wp_page  
7          0.0451  vmlinux        vmlinux       _spin_unlock_irqrestore  
5          0.0322  vmlinux        vmlinux       __handle_mm_fault  
4          0.0258  libc-2.4.so    libc-2.4.so   malloc  
4          0.0258  libc-2.4.so    libc-2.4.so   mbrtowc  
1          0.0064  ld-2.4.so      ld-2.4.so     do_lookup_x
```

1	0.0064	libc-2.4.so	libc-2.4.so	_dl_addr
1	0.0064	libc-2.4.so	libc-2.4.so	_int_free
1	0.0064	libc-2.4.so	libc-2.4.so	mblen

In Linux on System z, OProfile incorporates only the kernel space callgraph support. The callgraph option is supported on Linux kernels Version 2.6 and later. When using the callgraph option, we can see what functions are calling other functions in the output report. OProfile will record the function stack every time that it takes a sample. To enable the callgraph support on OProfile, set the callgraph sample collection rate with a maximum depth. Use '0' to disable the callgraph functionality. See Example 3-20 for an example of starting call-graph.

Example 3-20 OProfile call-graph feature

```
opcontrol --start --callgraph=depth> --vmlinux=lib/modules/<Kernel version>
/build/vmlinux
```

Note: Currently, in Linux on System z, OProfile incorporates only the kernel space callgraph feature.

Example 3-21 shows an example of the OProfile callgraph report.

Example 3-21 Example of OProfile Call-graph

```
lnx02:~ # opreport --callgraph
CPU: CPU with timer interrupt, speed 0 MHz (estimated)
Profiling through timer interrupt
```

samples	%	image name	app name	symbol name

15272	48.0720	lib_app1	lib_app1	main
15272	100.000	lib_app1	lib_app1	main [self]

14281	44.9526	vmlinux	vmlinux	.text
14281	100.000	vmlinux	vmlinux	.text [self]

1694	5.3322	no-vmlinux	no-vmlinux	(no symbols)
1694	100.000	no-vmlinux	no-vmlinux	(no symbols) [self]

147	0.4627	vmlinux	vmlinux	cpu_idle
147	100.000	vmlinux	vmlinux	cpu_idle [self]

65	0.2046	libc-2.4.so	libc-2.4.so	_int_malloc
65	100.000	libc-2.4.so	libc-2.4.so	_int_malloc [self]

48	0.1511	vmlinux	vmlinux	get_page_from_freelist

48	100.000	vmlinux	vmlinux	get_page_from_freelist
[self]				
39	0.1228	vmlinux	vmlinux	page_remove_rmap
39	100.000	vmlinux	vmlinux	page_remove_rmap [self]

3.1.11 zFCP statistics

With zFCP statistics, we can record FCP LUN I/O activities as statistical information. With this feature we can collect various statistical data such as I/O request data sizes or I/O latencies. See Figure 3-2 for an overview of zFCP statistics.

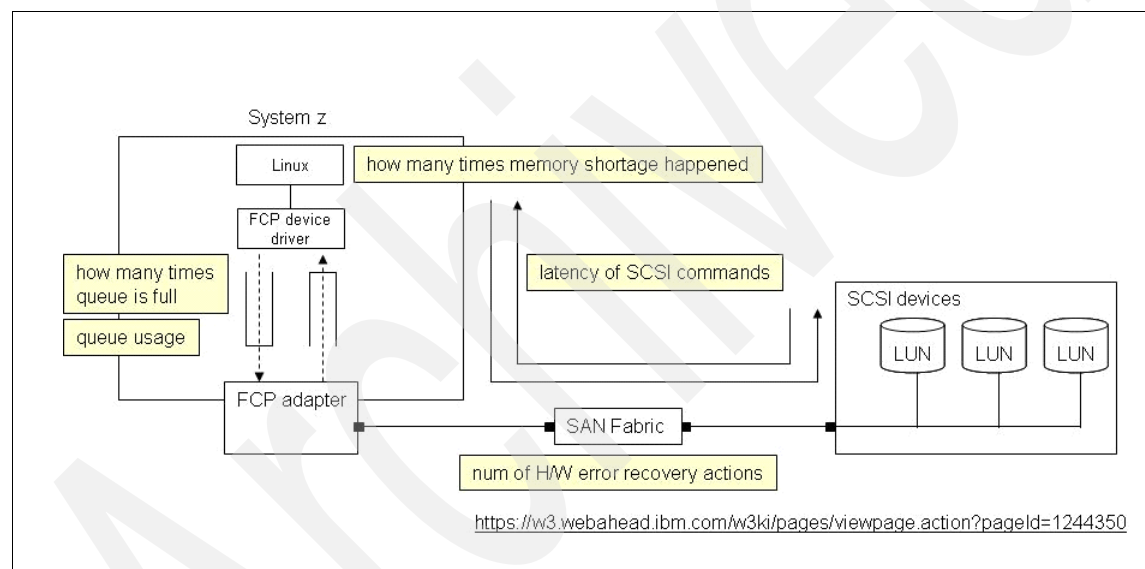


Figure 3-2 zFCP statistics overview

Statistical data on FCP devices can be collected starting with SUSE Linux Enterprise Server 9 SP3 + maintenance (kernel Version 2.6.5-7.283 and later) and SUSE Linux Enterprise Server 10 GA (kernel Version 2.6.16.21-0.8 and later).

Gathering zFCP statistics

To gather zFCP statistical information:

1. Depending on your Linux distribution, the files for zFCP statistics can be found as follows:
 - For SLES10 and later, depending on where debugfs is mounted, <mount_point_debugfs>/statistics. For example, if debugfs is mounted at directory /sys/kernel/debug/, all the collected statistics data can be found at /sys/kernel/debug/statistics/.
 - For SLES9, depending on where proc is mounted (SLES9 does not use debugfs), <mount_point_proc>/statistics.
2. For each device (adapter as well as LUN) a subdirectory is created when mounting the device. A subdirectory is named:
 - zfc<device-bus-id> for an adapter
 - zfc<device-bus-id>-<WWPN>-<LUN> for a LUN
3. Each subdirectory contains two files, a data file and a definition file.

To switch on data gathering for the devices, see Example 3-22

Example 3-22 Enabling zFCP statistics

```
echo on=1 > definition
```

To switch off data gathering for the devices, see Example 3-23.

Example 3-23 Disabling zFCP statistics

```
echo on=0 > definition
```

Table 3-1 shows an example of the output formats of different statistic types.

Table 3-1 Example statistics types

Statistics time	Output format	Number of lines
Value	<name> <total>	1
range	<name> <total> <min> <avg> <max>	1
list	<name> <Xn> <total for Xn>	<= entries_max

Statistics time	Output format	Number of lines
array	<name> "<="<Xn> <total for interval> <name> ">"<Xm> <total for interval>	intervals as determined by base_interval, scale, range_min, range_max
history (mode=increments, mode=products)	<name> <time-stamp> <total>	<= entries_max
history (mode=range)	name> <time-stamp> <total> <min> <avg> <max>	<= entries_max
raw	<name> <time-stamp> <serial> <X> <Y>	<= entries_max

The breakdown of latency (which is the minimum time required to move data from one point to another) when using SCSI can be seen in Figure 3-3.

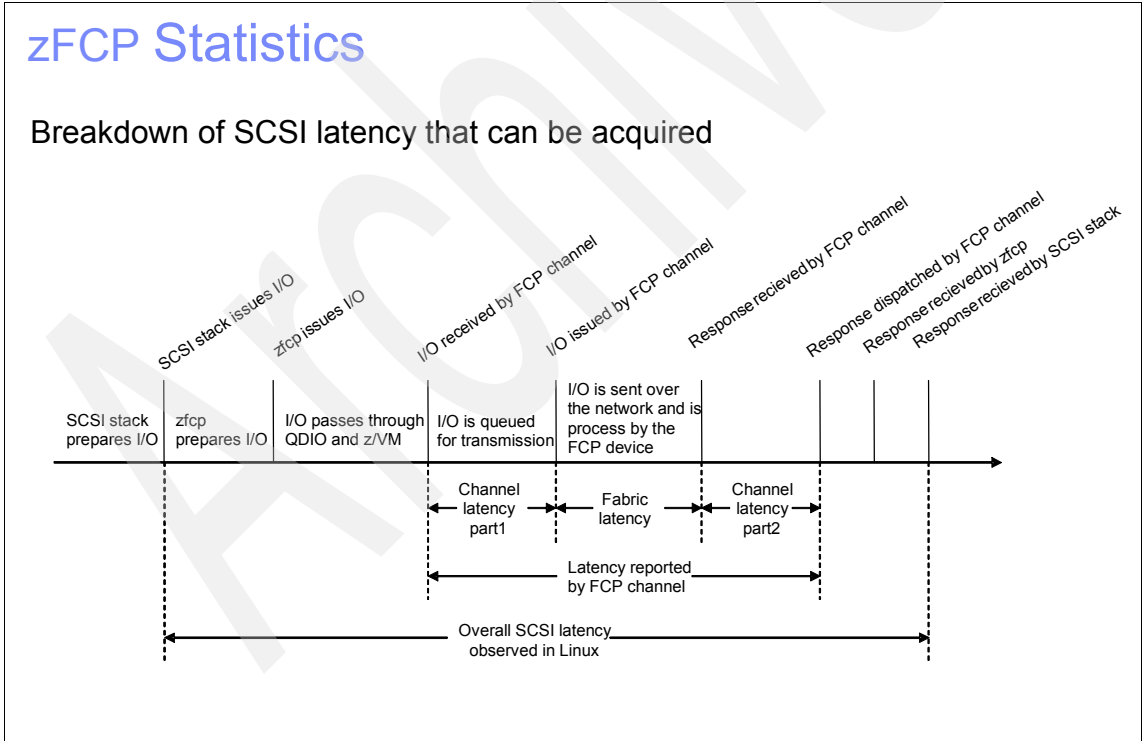


Figure 3-3 SCSI latency breakdown

A sample of SCSI latency statistics can be seen in Example 3-24.

Example 3-24 Example SCSI latency statistics

```
cat /proc/statistics/zfcp-0.0.3c00-0x5005076300c20b8e-0x5500000000000000/data
request_sizes_scsi_write >4096 339
request_sizes_scsi_read <=0 0
request_sizes_scsi_read <=512 0
request_sizes_scsi_read <=1024 0
request_sizes_scsi_read <=1536 0
request_sizes_scsi_read <=2048 0
request_sizes_scsi_read <=2560 0
request_sizes_scsi_read <=3072 0
request_sizes_scsi_read <=3584 0
request_sizes_scsi_read <=4096 16259
request_sizes_scsi_read >4096 0
request_sizes_scsi_nodata <=0 0
request_sizes_scsi_nodata <=512 0
request_sizes_scsi_nodata <=1024 0
:
:
:
request_sizes_scsi_write 137
request_sizes_scsi_read 14308
request_sizes_scsi_nodata 0
request_sizes_scsi_nofit 0
request_sizes_scsi_nomem 0
request_sizes_timedout_write 0
request_sizes_timedout_read 0
request_sizes_timedout_nodata 0
latencies_scsi_write 137
latencies_scsi_read 14308
latencies_scsi_nodata 0
pending_scsi_write 137
pending_scsi_read 14309
occurrence_erp 0
```

3.1.12 Summary of information-gathering tools

For your convenience, we provide a summary of tools that you can use in your Linux environment to gather information (Table 3-2).

Table 3-2 Summary of Linux-based information-gathering tools

Tools	Description	CPU	Memory	Disk	Network
vmstat	Resource utilization can be acquired at specified intervals of time.	X	X	X	
mpstat	CPU utilization can be acquired at specified intervals of time.	X			
sar	Resource utilization can be acquired at specified intervals of time.	X	X	X	X
top	Report on per-process CPU and memory utilization.	X	X		
ps	Report on process information, status, and resource utilization.	X	X		
free	Report on memory and swap utilization.	X	X		
ipcs	Report on utilization of the shared memory, the semaphore, and the message queue.		X		
iostat	Provides information about I/O activities of device.	X		X	
netstat	Reports network connection and statistical information.				X
OProfile	Reports statistical information of the functions executed.	X			
DASD Statistics	Provides I/O statistic on DASD devices.			X	
zFCP statistics	Provides I/O statistics on each FCP LUN.			X	

3.2 Dump and trace

The system **dump** command copies selected kernel structures to a dump file or device when an unexpected system error occurs. The dump device can be

dynamically configured, which means that either the tape or DASD devices can be used to receive the system dump.

A system kernel dump contains all the vital structures of the running system, such as the process table, the kernel's global memory segment, and the data and stack segment of each process. When we examine system data that maps into these structures, we can gain valuable kernel information that can explain why the dump was called. Also, the dump technique can only take a snapshot of the kernel structure at a given point of time.

On the other hand, with the tracing mechanism we would be able to record system calls and other application-based issues dynamically.

3.2.1 When to use dump

When an application fails and halts execution or whenever a system crashes with a kernel panic, we recommend doing a dump, since with a dump we can record and analyze the execution and memory snapshot of the system when the issue happen. See Figure 3-4.

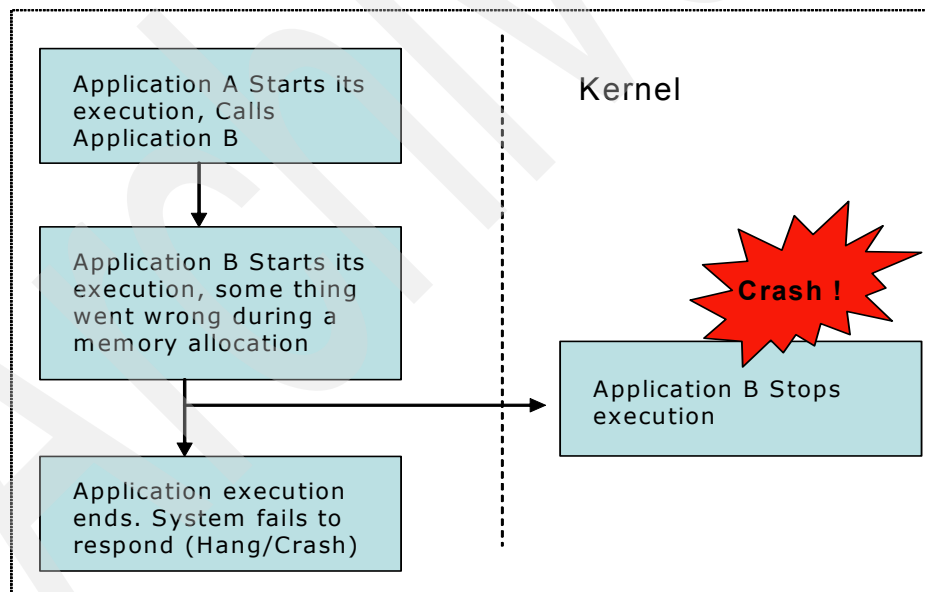


Figure 3-4 Example of when to use dumps

3.2.2 When a trace should be run

Unlike Figure 3-4 on page 81, there are situations in which the application would be still in execution, but does not respond. In this case we recommend using trace facilities, rather than the dump.

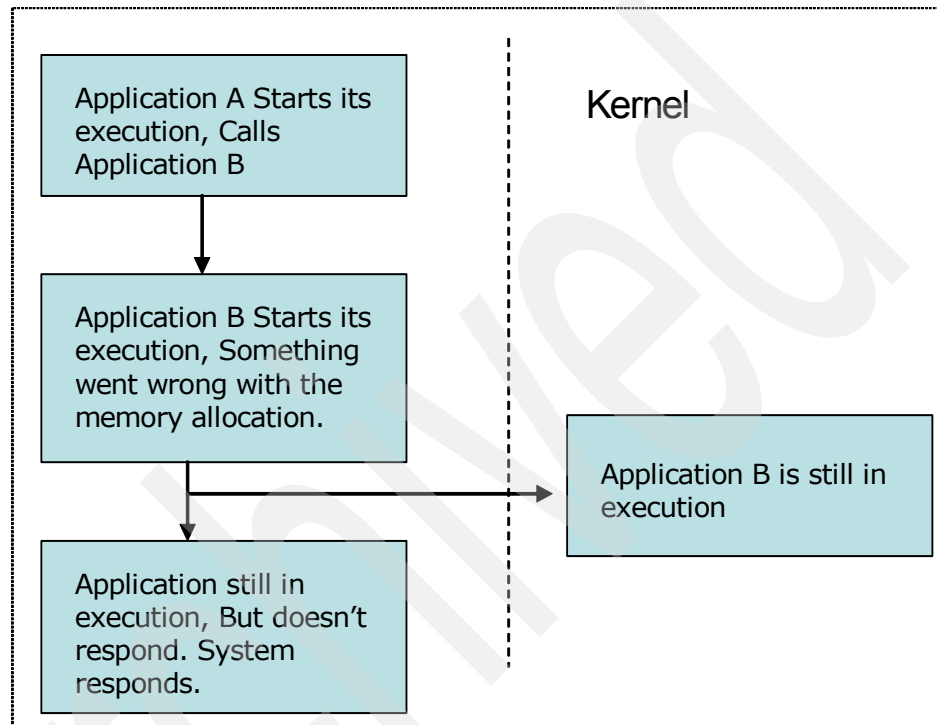


Figure 3-5 Example of when to use trace facilities

If we use dump, it would be only a snapshot. So it would be required to use a series of dumps at various execution points. Instead, trace facilities would provide us with information about system calls and signal traces to the standard error devices.

3.2.3 Stand-alone dump

A stand-alone dump is used to collect diagnostic information about the entire system. After a stand-alone dump is taken, the system cannot resume normal processing—an IPL is required. Stand-alone dump is very useful in investigating system hangs, kernel memory, and system call status.

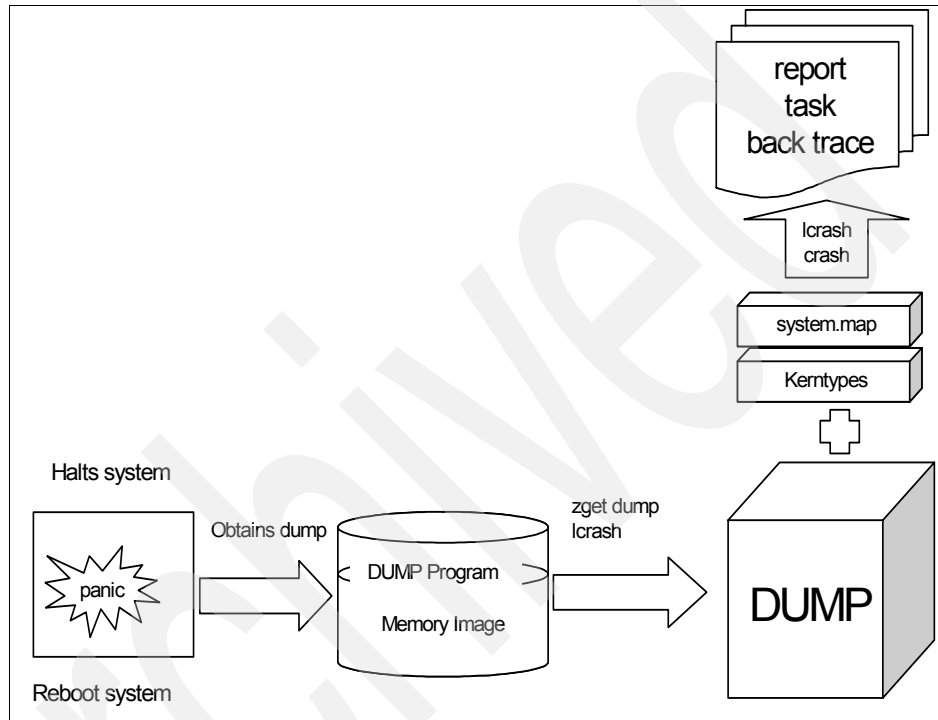


Figure 3-6 Stand-alone dump

Three stand-alone dump tools are shipped in the s390-tools package as part of the ziplt package:

- ▶ DASD dump tool for dumps on DASD volumes
- ▶ Tape dump tool for dumps on (channel-attached) tape devices
- ▶ SCSI disk dump tool for dumps on a SCSI disks

To create a stand-alone dump:

1. Create a dump device for dumping the memory. We recommend that you have device storage allocation around memory size plus approximately 10 MB space. You can define the dump device in the ziplt.conf file found in the /etc directory.

2. To initiate a dump, we must IPL the dump device. This is destructive. That is, the running Linux operating system is killed.
3. Reboot the original Linux images and mount the already created dump DASD.

3.2.4 Verification of the dump

After the dump device is mounted on the system, we need to inspect the dump to identify the problem, also referred as post-processing of the dump device. In Linux we have tools to validate the dump devices or files.

zgetdump

Use the `zgetdump` tool to read the given dump device and write its contents to standard output or redirect to a file.

Example 3-25 Convert a dump image into a file using `zgetdump`

```
zgetdump /dev/dasdXX > dump.X
```

lcrash

Validation and inspection of the dump can also be done using the `lcrash`—the Linux crash dump analyzer.

Example 3-26 Verification of the dump file using `lcrash`

```
lcrash -i -d /dev/dasdXX
```

To copy a dump from DASD to a file system, call `lcrash` with the `-s` (save dump) and `-d` (dump device) options. The `-s` option specifies the destination directory for the resulting dump file (dumpfile).

Example 3-27 Converting a dump image to a file using `lcrash`

```
lcrash -d /dev/dasdx1 -s dumpfile
```

Note: During verification of the dump images, the tools should display the message `Dump End Marker found: this dump is valid`. This means that the dump is a valid dump that can be used for further processing.

Creating reports from stand-alone dump using `lcrash`

Install `lkcdutils` if `lcrash` is not found on the system.

For creating a report on the dump file, issue the commands shown in Example 3-28.

Example 3-28 Commands to issue

```
lcrash -r System.map dumlin Kerntypes > report.txt
```

From the report we would be able to get the following information:

- ▶ Dump summary
- ▶ Log buffer
- ▶ Task list
- ▶ Information of processes assigned to CPUs

Figure 3-7 shows a lcrash dump summary.

```
=====
LCRASH CORE FILE REPORT
=====
GENERATED ON:
    Mon Mar 1 18:11:54 2008

TIME OF CRASH:
    Mon Mar 1 15:34:02 2008

PANIC STRING:
    zSeries-dump (CPUID = ff08212320640000)

MAP:
    /boot/System.map-2.6.5-7.282-s390x
DUMP:
    dump
KERNTYPES:
    /boot/Kerntypes-2.6.5-7.282-s390x

=====
COREFILE SUMMARY
=====
    The system died due to a software failure.
=====
UTSNAME INFORMATION
=====
    sysname :
    nodename :
    release :
    version :
    machine : s390x
```

Figure 3-7 Example dump summary of lcrash report

As seen in Figure 3-7, the summary says that the system crashed because of a software failure. Likewise, the report consists of other information, as listed above, which can be useful for problem determination at a high level.

Figure 3-8 shows the task list of an lcrash report. This can used to see all the process that are running during that point of time.

```
=====
CURRENT SYSTEM TASKS
=====
```

ADDR	UID	PID	PPID	STATE	FLAGS	CPU	NAME
0x400000	0	0	0	0	0	0	swapper
0x7177b0	0	1	0	1	0x100	-	init
0xf727d0	0	2	1	1	0x40	-	migration/0
0xf72020	0	3	1	1	0x8040	-	ksoftirqd/0
0x9927e0	0	4	1	1	0x8140	-	events/0
0x992030	0	5	4	1	0x8040	-	khelper
0xf8477f0	0	6	4	1	0x8040	-	kblockd/0
0xf863810	0	60	4	1	0x8040	-	cio
0xf863060	0	61	4	1	0x8040	-	cio_notify
0xf86e820	0	62	4	1	0x8040	-	kslowcrw
0xf85d050	0	136	4	1	0x8040	-	apldata
0xf8a4000	0	138	4	1	0xa040	-	pdflush
0xf8bc080	0	139	4	1	0xa040	-	pdflush
0xf8b17f0	0	141	4	1	0x8040	-	aio/0
0xf8b7060	0	140	1	1	0x40840	-	kswapd0
0xf8f5090	0	198	1	1	0x40	-	kmcheck
0xf8ad020	0	357	1	1	0x40	-	kjournald
0xf86e070	0	358	1	1	0x140	-	init
0xee607d0	0	359	358	1	0x100	-	boot
0xee60020	0	431	359	1	0x100	-	S07boot.localfs
0xea67800	0	449	431	1	0x100	-	sulogin

Figure 3-8 Example task list in the report

Also, the log buffer would provide us with useful information about the log messages that do not appear on the console or dmesg. But we recommend sending in the dumpfile, system map, kerntypes, and the lcrash report to IBM Support Services for further investigation.

3.2.5 Core dump

A core dump records the state of the process at the time of the crash. On most Linux distributions, core dumps are disabled. Once we have the core dump

enabled on the system, the next time an application crashes, Linux is able to write the core to a file.

As an example, we executed a simple program that would keep on forking new processes. At any one point the Linux would be running out of process ID and resource, in which case the application would not be responding and would be busy waiting for other process to finish so that it can span more processes. This is a classic example of a core dump, since we know the PID of the process and we can obtain the core dump for the process for further investigation.

Once the program started creating processes, we captured the process ID (PID) for the program using the `ps -ef` command. At this point the application is not responding and the Linux system is also low on resources.

In Figure 3-9, we will take an example PID and obtain the core dump for it. Then in the debugging section we will investigate the core dumps created using different tools like gdb, ltrace, and strace.

```
lrx02~ # ps -ef
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	Mar23	?	00:00:01	init [3]
root	2	1	0	Mar23	?	00:00:00	[migration/0]
root	3	1	0	Mar23	?	00:00:00	[ksoftirqd/0]
root	4	1	0	Mar23	?	00:00:00	[migration/1]
root	5	1	0	Mar23	?	00:00:00	[ksoftirqd/1]
postfix	1713	1186	0	05:02	?	00:00:00	pickup -l -t fifo -u
root	1806	1099	0	06:17	?	00:00:00	sshd: root@pts/0
root	1809	1806	0	06:17	pts/0	00:00:00	-bash
root	1830	1099	0	06:18	?	00:00:00	sshd: root@pts/1
root	1834	1830	0	06:18	pts/1	00:00:00	-bash
root	1863	1834	0	06:20	pts/0	00:00:00	./a.out
root	1864	1863	0	06:20	pts/0	00:00:00	./a.out

Figure 3-9 Process listing

System and environment setup for core dump

Before we can go ahead with generating a core dump, we need to enable the core dump in the system. This can be done using the **ulimit -c unlimited** command. This command would set the core enabled for the current shell and all the processes started from it.

```
lnx02:/etc/security # ulimit -c unlimited
lnx02:/etc/security # ulimit -a
core file size          (blocks, -c) unlimited
data seg size          (kbytes, -d) unlimited
file size              (blocks, -f) unlimited
pending signals        (-i) 4096
max locked memory      (kbytes, -l) 32
max memory size        (kbytes, -m) unlimited
open files             (-n) 1024
pipe size              (512 bytes, -p) 8
POSIX message queues   (bytes, -q) 819200
stack size             (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes     (-u) 4096
virtual memory         (kbytes, -v) unlimited
file locks             (-x) unlimited
```

Figure 3-10 Example enabling of core file size using ulimit

For individual users we can set the values through the `/etc/security/limits.conf` or `ulimits.conf` file. The same thing can also be achieved by adding the “`ulimit -c unlimited`” in the `.bash_profile`.

For system-wide setting up of the core values, we can also place the core setup information in `/etc/initscript`, but it has to be handled carefully for unexpected core dumps in the system.

Normally, the core file would be generated in the current directory, but we can also set up the output location of the core file. This can be done by using the command **sysctl**, or we can also permanently added the configuration in the `sysctl.conf` file under the `/etc` directory.

Example 3-29 Example to set up output location using sysctl

```
sysctl -w kernel.core_pattern=/<output_dir>/<file name : core.%h.%t.%e>
```

Obtaining a core dump

There are multiple ways to obtain a core dump in Linux.

SIGNAL

Send a signal to the process using the **kill** command. The PID of the application can be gathered from the 'ps -ef' process listing. As per the POSIX standard, the following (Example 3-30) are the some of the important signals available in Linux. We can use any of the signals that has an action of creating a core dump.

Example 3-30 Important signals available in Linux

Signal	Value	Action	Description
SIGINT	2	TERM	Interrupt from keyboard
SIGQUIT	3	CORE	Quit from keyboard
SIGILL	4	CORE	Illegal instruction
SIGABRT	6	CORE	Abort signal
SIGFPE	8	CORE	Floating point exception
SIGKILL	9	TERM	Kill signal
SIGSEGV	11	CORE	Invalid memory reference
SIGPIPE	13	TERM	Broken pipe signal
SIGTERM	15	TERM	Termination signal

In Example 3-31, we would issue the SIGABRT (abort signal) signal for obtaining a core dump.

Example 3-31 Sending a signal to a process using kill

```
kill -6 <PID>  
Kill -6 1863
```

gdb

We can use `gdb` to create a core dump. For creating a core dump of a particular process, we specify the problematic process PID as an debug parameter, and once it is in the debug mode, from `gdb` we can obtain a core dump. In Figure 3-11, `gdb` has created a core dump with the name `core.1863` in the current directory.

```
lnx02:~ # gdb -p 1863
GNU gdb 6.3
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License,
and you are
welcome to change it and/or distribute copies of it under certain
conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for
details.
This GDB was configured as "s390x-suse-linux".
Attaching to process 1863
Reading symbols from /root/a.out...done.
Using host libthread_db library "/lib64/tls/libthread_db.so.1".
Reading symbols from /lib64/tls/libc.so.6...done.
Loaded symbols for /lib64/tls/libc.so.6
Reading symbols from /lib/ld64.so.1...done.
Loaded symbols for /lib/ld64.so.1
0x000002000000cc1d4 in wait () from /lib64/tls/libc.so.6
(gdb) generate-core
Saved corefile core.1863
(gdb) quit
The program is running. Quit anyway (and detach it)? (y or n) y
Detaching from program: /root/a.out, process 1863
```

Figure 3-11 Example of obtaining core dump using `gdb`

gcore

Like `gdb`, in Red Hat we have a `gcore` as an default command to creates a core image of the specified process, suitable for use with `gdb`. By default, the core is written to the file `core.<pid>` in the current directory. The process identifier, `pid`, must be given on the command line.

Example 3-32 Example creation of core using `gcore`

```
gcore <PID>
gcore 1863
```

3.2.6 Summary of dump techniques

Stand-alone dumps are effective in cases of system failures like crash or hang. With stand-alone dumps, we are able to:

- ▶ Analyze kernel panic, oops messages, and so on.
- ▶ Identify direct reasons to lead system down.
- ▶ Identify what tasks are processed at failure time.
- ▶ Analyze a root cause with memory usage.

On the other hand, core dumps are useful for problem determination of abnormal termination of user space programs and to:

- ▶ Identify the causes of the problem that process is exhibiting.
- ▶ Analyze the memory structures of the process during a hang or abnormal termination.

Table 3-3 Summary of dump techniques.

Technique	Used for	Description
Stand-alone dump	Kernel-related problems, like system hangs	Physical memory dump
Core dump	Process problem	Process memory dump. Used with gdb, Sysrq, and so on.

3.3 Debugging tools

Linux-based tools are the best choices for debugging application, kernel, or process-related problems. In this section we discuss important tools that are used often for problem determination.

As per the problem determination methodology, first the investigation on the problem has to be start on the application layer. If the investigation reveals that the application layer is not the cause, we further move down the ladder to the OS hardware layers, as appropriate.

Debugging tools are of prime importance while investigating problems such as:

- ▶ System hangs
- ▶ Kernel-related problems or system call status
- ▶ Kernel memory related issues

The basic and easiest of all the debugging techniques is to start adding print statements to the application code. Upon execution of the application, we can analyze the outputs from the print statements through which the cause of the problem can be narrowed down. Also, we would be able to immediately see whether the application layer is causing the problem. If the application is causing the issue, by using the print statements we would be able to narrow down the area of the application that causes the problem and take appropriate action.

3.3.1 gdb

The GNU debugger (gdb) allows us to examine the internals of another program while the program executes or retrospectively to see what a program was doing at the moment that it crashed. The gdb also allows us to examine and control the execution of code and is very useful for evaluating the causes of crashes or general incorrect behavior. gdb provides a text-based user interface, and it can be used to debug programs written in several languages (C, C++, and others). gdb can be used to perform the following operations, which are helpful in the process of debugging a compiled program:

- ▶ Setting up break points
- ▶ Displaying program values and attributes
- ▶ Execution step through line by line
- ▶ Information about the stack frame

gdb is run from the shell with the command **gdb** with the program name as a parameter, or we can use the file command once inside gdb to load a program for debugging. Both of these assume that you execute the commands from the same directory as the program. Once the executables' symbols are loaded, there are three ways to view the process with the debugger:

- ▶ View the running process using the **attach** command.
- ▶ Start the program using the **run** command.
- ▶ Use the core file to view the state of the process when it crashed.

In Example 3-33 we executed a simple program that does an illegal memory reference and fails with a segmentation fault.

Example 3-33 Program execution that fails with a segmentation fault

```
nx02:~ # ./segfault
struct1->i : 40
struct1->j : 50
Segmentation fault
```

Now we execute the simple program using gdb and analyze the output using a few of the options found in gdb. In Example 3-34, when the program receives a segmentation fault, gdb returns to its prompt. It also lists the place where the segmentation fault occurs with the line number. A closer look reveals that the program is trying to initialize or read an unknown memory location. Also, from the gdb output it is clear that the info locals and list commands get information about the currently selected stack frame.

Example 3-34 Example for viewing the process using gdb

```
lnx02:~ # gdb segfault
GNU gdb 6.3
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and
you are
welcome to change it and/or distribute copies of it under certain
conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for
details.
This GDB was configured as "s390x-suse-linux"...Using host libthread_db
library "/lib64/tls/libthread_db.so.1".

(gdb) run
Starting program: /root/segfault
struct1->i : 40
struct1->j : 50

Program received signal SIGSEGV, Segmentation fault.
0x0000000080000616 in main () at segfault.c:17
17     printf (" non_struct->i : %d \n", non_struct->i);
(gdb) backtrace
#0  0x0000000080000616 in main () at segfault.c:17
(gdb) list
17     printf (" non_struct->i : %d \n", non_struct->i);
18     printf (" non_struct->j : %d \n", non_struct->j);
19     }
(gdb) info locals
temp = {i = 40, j = 50}
struct1 = (struct some_struct *) 0x3fffffff1c8
non_struct = (struct some_struct *) 0x233220534d
(gdb) quit
The program is running. Exit anyway? (y or n) y
lnx02:~ #
```

Analyzing a core file using gdb

As discussed in the previous section, we obtained a core dump for the segfault program (Example 3-35).

Example 3-35 Obtaining a core dump

```
lnx02:~ # ulimit -c unlimited
lnx02:~ # ./segfault
struct1->i : 40
struct1->j : 50
Segmentation fault (core dumped)
```

This core file contains a complete copy of the pages of memory used by the program at the time at which it was terminated. Incidentally, the term *segmentation fault* refers to the fact that the program tried to access a restricted memory *segment* outside the area of memory that had been allocated to it.

Now we invoke the GNU debugger to load the core files for debugging the process. To investigate the cause of the crash we display the value of the pointer `non_struct->i` using the `print` command. See Example 3-36.

Example 3-36 Loading the core file to gdb

```
t2930030:~ # gdb segfault core
GNU gdb 6.3
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and
you are
welcome to change it and/or distribute copies of it under certain
conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for
details.
This GDB was configured as "s390x-suse-linux"...Using host libthread_db
library "/lib64/tls/libthread_db.so.1".

Core was generated by `./segfault'.
Program terminated with signal 11, Segmentation fault.
Reading symbols from /lib64/tls/libc.so.6...done.
Loaded symbols for /lib64/tls/libc.so.6
Reading symbols from /lib/ld64.so.1...done.
Loaded symbols for /lib/ld64.so.1
#0 0x0000000080000616 in main () at segfault.c:17
17     printf (" non_struct->i : %d \n", non_struct->i);
(gdb) backtrace
#0 0x0000000080000616 in main () at segfault.c:17
```

```
(gdb) print non_struct->i
Cannot access memory at address 0x233220534d
(gdb) quit
```

In Example 3-36 on page 95, when gdb tries to print the variable, it displays a message that it is not able to access the memory location. That is the reason for the segmentation fault.

Note: For more in-depth information about gdb, refer the gdb manual available at:

<http://sourceware.org/gdb/documentation/>

3.3.2 strace

The **strace** command is a powerful debugging tool available for the Linux on System z architecture. The strace utility intercepts and records the system calls that are called by a process and the signals that are received by a process. The name of each system call, its arguments, and its return value are printed on standard error or to the file specified with the -o option. Using strace leads to the shortening of analysis time during problem determination of user space programs as well as system commands. The strace utility can be used for debugging in two methods:

- ▶ Start the program or application with the strace utility (Example 3-37).

Example 3-37 Example for obtaining strace for ls command

```
strace -o out_file -tt -f ls
```

- ▶ Obtain the strace of a process that is already running (Example 3-38).

Example 3-38 Strace for PID 1834

```
strace -o out_file -tt -f -p 1834
```

The traces of only a specified system calls can also be obtained by passing a certain option, but we always recommend obtaining the traces of all system calls when the problem area is not narrowed down. In Example 3-39, strace would only trace the signal system calls for the process.

Example 3-39 Obtain only signal system calls

```
strace -o out_file -tt -ff -e trace=signal -x ./segfault
```

It is also possible to obtain only the trace of the open() system call using the option “-e trace=open.

For our demonstration of the strace, we wrote a simple program that tries to open a file descriptor that has not been initialized at all. Even though the program does not fail, using strace, we would be debug and see that the program has some flaws. In Example 3-40 strace points out that the program tries to open a file that is not available.

Example 3-40 Debugging a invalid file descriptor problem using strace

```
lnx02:~ # strace -o report.txt ./unmemory
Could not open the file
lnx02:~ # cat report.txt
execve("./unmemory", ["/unmemory"], [/* 53 vars */]) = 0
uname({sys="Linux", node="lnx02", ...}) = 0
brk(0) = 0x80002000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x2000001a000
open("/etc/ld.so.preload", O_RDONLY) = -1 ENOENT (No such file or
directory)
open("/etc/ld.so.cache", O_RDONLY) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=90768, ...}) = 0
mmap(NULL, 90768, PROT_READ, MAP_PRIVATE, 3, 0) = 0x2000001b000
close(3) = 0
open("/lib64/tls/libc.so.6", O_RDONLY) = 3
read(3, "\177ELF\2\2\1\0\0\0\0\0\0\0\0\0\0\3\0\26\0\0\0\1\0\0\0"...
, 640) = 640
lseek(3, 624, SEEK_SET) = 624
read(3, "\0\0\0\4\0\0\0\20\0\0\0\0\1GNU\0\0\0\0\0\0\0\2\0\0\0\6"...
, 32) = 32
fstat(3, {st_mode=S_IFREG|0755, st_size=1542487, ...}) = 0
mmap(NULL, 1331616, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) =
0x20000032000
madvise(0x20000032000, 1331616, MADV_SEQUENTIAL|0x1) = 0
mmap(0x20000157000, 118784, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED, 3, 0x124000) = 0x20000157000
mmap(0x20000174000, 12704, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x20000174000
close(3) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x20000178000
munmap(0x2000001b000, 90768) = 0
brk(0) = 0x80002000
brk(0x80023000) = 0x80023000
brk(0) = 0x80023000
```

```

open("file.txt", O_RDONLY)          = -1 ENOENT (No such file or
directory)
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1), ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x2000001b000
write(1, "Could not open the file \n", 25) = 25
munmap(0x2000001b000, 4096)         = 0
exit_group(0)                       = ?

```

3.3.3 ltrace

The ltrace utility intercepts and records the dynamic library calls from the executed process and the signals that are received by that process. It can also intercept and print the system calls executed by the program.

As with strace, ltrace is also very similar to obtaining the traces. The ltrace utility can be used for debugging with two methods:

- Invoke the user program or application with the ltrace utility (Example 3-41).

Example 3-41 Example for obtaining ltrace for ls command

```

lnxo2:~ # ltrace -o out_file -tt -f echo .hello.
.hello.
lnxihs:~ # cat out_file
9901 21:02:44.476858 __libc_start_main(0x800014b0, 2, 0x3ffffa0f348,
0x80003344, 0x80003340 <unfinished ...>
9901 21:02:44.477045 getenv("")
= "SIXLY_CORRECT"
9901 21:02:44.477126 setlocale(6, "")
= "\247\364\375\203\343\2610"
9901 21:02:44.477687 bindtextdomain("/usr/share/locale",
"/usr/share/locale") = "/usr/share/locale"
9901 21:02:44.477791 textdomain("/usr/share/locale")
= "coreutils"
9901 21:02:44.477877 __cxa_atexit(0x80001a84, 0, 0, 0x20000065610,
0x80003828) = 0
9901 21:02:44.477957 getopt_long(2, 0x3ffffa0f348, "", 0x80005028,
NULL) = -1
9901 21:02:44.478044 fputs_unlocked(0x3ffffa0f638, 0x2000018ba80,
0x8000162c, 1, 1) = 1
9901 21:02:44.478152 __overflow(0x2000018ba80, 10, 0x8000157c,
0x20000027007, 0) = 10
9901 21:02:44.478335 exit(0 <unfinished ...>

```

```

9901 21:02:44.478378 __fpending(0x2000018ba80, 0, 0x80001a84,
0x2000018a8a8, 0) = 0
9901 21:02:44.478457 fclose(0x2000018ba80)
= 0
9901 21:02:44.478631 +++ exited (status 0) +++

```

- Hook the PID of a process already running with the ltrace utility (Example 3-42).

Example 3-42 strace for PID 1834

```
ltrace -o out_file -tt -f -p 10158
```

Note: ltrace would not be able to trace threaded processes. If a process generates a thread that shares the same address space (that is, calling the *clone* system call with parameters *CLONE_VM* and *CLONE_THREAD*), the traced process will die with SIGILL.

Another simple example illustrated here is obtaining the system call statistics, like count time and calls using ltrace. In Example 3-43 we passed the *ls* command as the executable to be traced. From Example 3-43 we can see which function is called most of the time or consumes lots on CPU cycles. By this the developers can optimize there code to run efficiently.

Example 3-43 Call tracing using ltrace

```

lnxihs:~ # ltrace -c ls
autoinst.xml forker netperf out_file segfault tcp_crr tcp_stream
threder vpd.properties
bin kerntest netserver rmfpms segfault.c tcp_rr test.c
threder.c
% time seconds usecs/call calls function
-----
34.01 0.004400 4400 1 qsort
15.20 0.001966 19 101 __errno_location
15.13 0.001958 19 102 __ctype_get_mb_cur_max
7.98 0.001033 22 45 __overflow
7.10 0.000919 19 47 strcoll
5.19 0.000671 20 32 readdir
4.13 0.000534 534 1 setlocale
3.49 0.000452 19 23 malloc
2.69 0.000348 20 17 fwrite_unlocked
1.34 0.000173 21 8 getenv
0.58 0.000075 75 1 opendir
0.44 0.000057 19 3 free

```

0.42	0.000054	54	1 fclose
0.30	0.000039	39	1 isatty
0.26	0.000033	33	1 closedir
0.25	0.000032	32	1 getopt_long
0.24	0.000031	31	1 ioctl
0.24	0.000031	31	1 bindtextdomain
0.19	0.000025	25	1 _setjmp
0.19	0.000024	24	1 _init
0.17	0.000022	22	1 realloc
0.16	0.000021	21	1 __fpending
0.15	0.000020	20	1 textdomain
0.15	0.000019	19	1 __cxa_atexit

100.00	0.012937	393	total

3.3.4 SysRq

SysRq is a key combo we can hit that the kernel will respond to regardless of whatever else it is doing, unless it is completely locked up. In case of any kernel problems other than a kernel oops or panic, a kernel core dump is not triggered automatically. If the system still responds to keyboard input to some degree, a kernel core dump can be triggered manually through a *magic SysRq* keyboard combination. For that, the kernel that is running on the system must be built with CONFIG_MAGIC_SYS-REQ enabled. These magic keystrokes give you a stack trace of the currently running processes and all processes, respectively. The systems would generate a /var/log/messages file for the back trace.

To enable the feature for the running kernel, issue the following commands. In Figure 3-12 we list possible values in `/proc/sys/kernel/sysrq`.

```
0 - disable sysrq completely
1 - enable all functions of sysrq
>1 - bitmask of allowed sysrq functions (see below for detailed
function description):
    2 - enable control of console logging level
    4 - enable control of keyboard (SAK, unraw)
    8 - enable debugging dumps of processes etc.
   16 - enable sync command
   32 - enable remount read-only
   64 - enable signalling of processes (term, kill, oom-kill)
  128 - allow reboot/poweroff
  256 - allow nicing of all RT tasks
```

Figure 3-12 Values that can be set for SysRq

Example 3-44 Enabling sysrq

```
echo 1 > /proc/sys/kernel/sysrq
```

To enable the magic SysRq feature permanently, update the `/etc/sysconfig/sysctl` file with `ENABLE_SYSRQ="yes"`. To obtain information about using the Sysrq, we can issue `"echo t > /proc/sysrq-trigger"`. The kernel traces would be written to `"/var/log/messages"`. See Example 3-45.

Example 3-45 SysRq report from /var/log/message

```
lnx02:/ # echo 1 > /proc/sys/kernel/sysrq
lnx02:/ # echo t > /proc/sysrq-trigger
lnx02:/ # tail /var/log/messages
Mar 25 13:54:34 lnx02 kernel: 00000000efc7bb40 00000000ffe8aa00
00000000fd092370 00000000fd092050
Mar 25 13:54:34 lnx02 kernel: 0000000000000000 00000000fd092050
000000000846d37b8 00000000efc7ba60
Mar 25 13:54:34 lnx02 kernel: 00000000ff401000 000000000035d1b8
000000000012a652 00000000efc7ba60
Mar 25 13:54:34 lnx02 kernel: 00000000001950d0 00000000fc6e1d00
07000000efc7bb78 07000000efc7bb78
Mar 25 13:54:34 lnx02 kernel: Call Trace:
```

```

Mar 25 13:54:34 lnx02 kernel: [<000000000013db4a>] schedule_timeout+0x9e/0x150
Mar 25 13:54:34 lnx02 kernel: [<00000000001b2080>] do_select+0x4a0/0x5e8
Mar 25 13:54:34 lnx02 kernel: [<00000000001b2514>] sys_select+0x34c/0x588
Mar 25 13:54:34 lnx02 kernel: [<000000000011f2ac>] sysc_noemu+0x10/0x16

```

Table 3-4 SysRq key assignments

KEY	Operation
'r'	Turns off keyboard raw mode and sets it to XLATE.
'k'	Secure Access Key (SAK) kills all programs on the current virtual console.
'b'	Will immediately reboot the system without syncing or unmounting your disks.
'o'	Will shut your system off.
's'	Will attempt to sync all mounted file systems.
'u'	Will attempt to remount all mounted file systems as read-only.
'p'	Will dump the current registers and flags to your console.
't'	Will dump a list of current tasks and their information to your console.
'm'	Will dump current memory information to your console.
'v'	Dumps the Voyager™ SMP processor information to your console.
'e'	Sends a SIGTERM to all processes, except for init.
'i'	Sends a SIGKILL to all processes, except for init.
'l'	Sends a SIGKILL to all processes, INCLUDING init.
'0-8'	Sets the console log level, controlling which kernel messages will be printed to your console. ('0', for example would make it so that only emergency messages like PANICs or OOPSes would make it to your console.)

Table 3-5 Sysrq Log levels

Level	explanation
0	KERN_EMERG system cannot be used
1	KERN_ALERT need action immediately
2	KERN_CRIT critical situation
3	KERN_ERR error situation
4	KERN_WARNING warning situation
5	KERN_NOTICE normal situation
6	KERN_INFO notification
7	KERN_DEBUG debug

Note: There are some reported problems that lead to a kernel panic when the SysRq key is used. The kernel has been fixed and the problems do not appear in the following releases:

- ▶ SLES9: 2.6.5–7.252 and later
- ▶ RHEL4: 2.6.9–35 and later

A stack trace from using SysRq would help in analyzing strange behaviors in the system such as when multiple user space programs stop responding and when keyboard input is still possible. Usually, a SysRq trace would be used in conjunction with other standalone and core dumps.

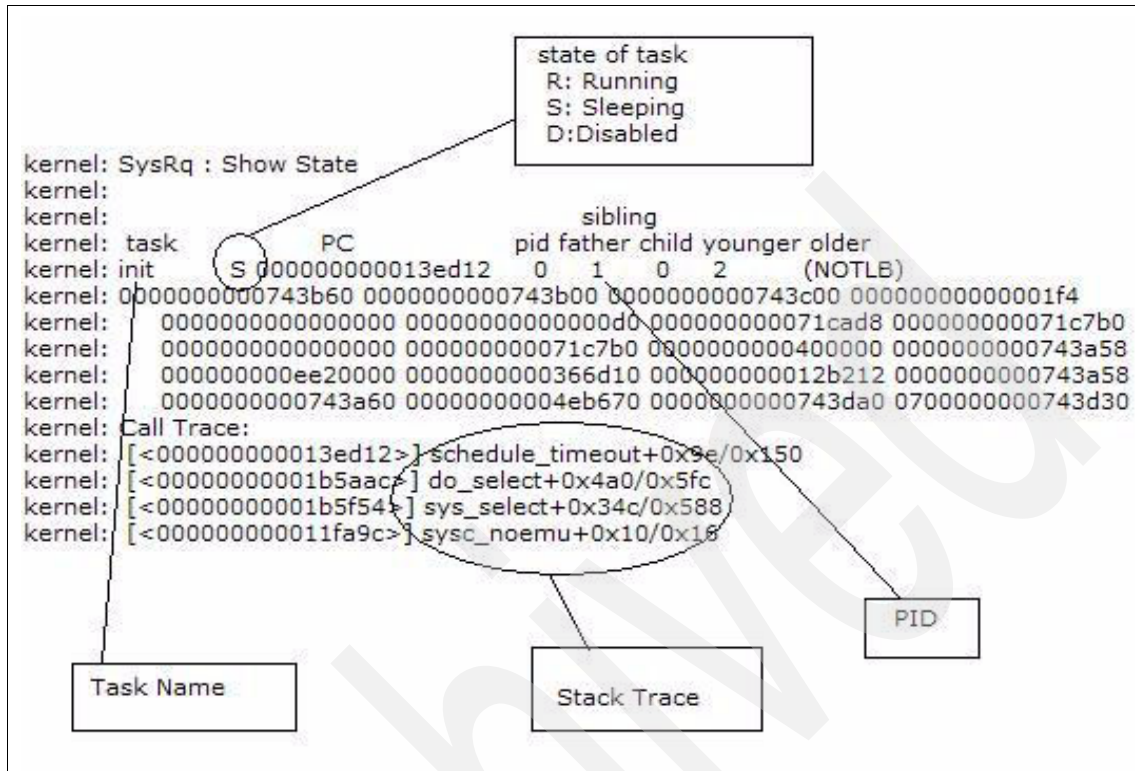


Figure 3-13 Example Sysrq report

3.3.5 s390dbf

The s390dbf debug feature traces kernel components such as device drivers. It produces trace logs, which can be useful when that specific component shows an error. Not all components produce the logs, however. A component must register with the feature to produce the traces.

The /proc directory holds debug logs from the s390dbf tool. Under the path /proc/s390dbf, there is a directory for each registered component. The names of the directory entries correspond to the names under which the components have been registered. There are three versions for the event (and exception) calls: one for logging raw data, one for text, and one for numbers. Each debug entry contains the following data:

- ▶ Timestamp
- ▶ CPU-Number of calling task
- ▶ Level of debug entry (0...6)
- ▶ Return address to caller

Select the file according to the area to be investigated and select the file under the /proc directory:

- ▶ /proc/s390dbf/cio_* : Common I/O
- ▶ /proc/s390dbf/qdio_* : qdio
- ▶ /proc/s390dbf/qeth_* : qeth
- ▶ /proc/s390dbf/zfcp_* : zfcp
- ▶ /proc/s390dbf/dasd :DASD
- ▶ /proc/s390dbf/0.0.xxxx : each DASD device

All debug logs have an actual debug level (ranging from 0 to 6). The default level is 3. Event and exception functions have a *level* parameter. The actual debug level can be changed with the help of the debugfs-filesystem through writing a number string "x" to the 'level' debugfs file that is provided for every debug log.

Example 3-46 s390dbf log level

```
lnx02:/ # cat /proc/s390dbf/qeth_trace/level
2
```

Now according to the level of tracing required, set the log value as shown in Example 3-47.

Example 3-47 Setting up log level

```
lnx02:/ # echo 6 > /proc/s390dbf/qeth_trace/level
lnx02:/ # cat /proc/s390dbf/qeth_trace/level
6
```

Once at the appropriate level of tracing, recreate the problem to save the s390dbf trace outputs.

Example 3-48 Saving the s390dbf trace to a file

```
cat /proc/s390dbf/qeth_trace/hex_ascii > output_file
```

Once the trace information from the s390dbf facility is saved, return to the old trace level (Example 3-49).

Example 3-49 Resetting the trace level back to default

```
lnx02:/ # echo 2 > /proc/s390dbf/qeth_trace/level
```

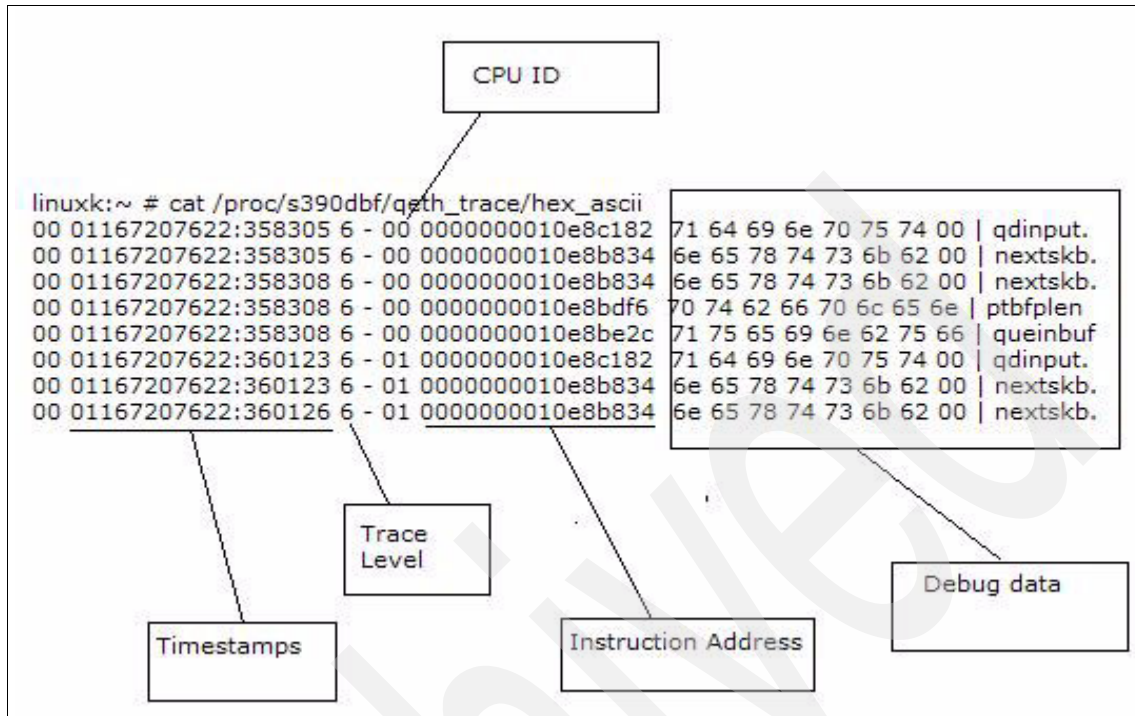


Figure 3-14 Resetting

s390dbf is useful for problem determination related to hardware devices, and also it can trace the communication between the device driver and the hardware device. With the s390dbf trace we can confirm that there is communication happening between with the devices (for example, I/O request to DASD device and status returned from it).

3.3.6 zFCP/SCSI

The zFCP trace feature enables the debugging of the zfcf device driver and the flow of the Fibre Channel protocol. zFCP trace should be used with the SCSI trace for any debugging.

The following trace information can be selected according to the problem target areas:

- ▶ loglevel_erp
- ▶ loglevel_fsf
- ▶ loglevel_fc
- ▶ loglevel_qdio

- ▶ loglevel_cio
- ▶ loglevel_scsi
- ▶ loglevel_other

In Example 3-50 we are enabling zFCP driver with the log level of loglevel_erp to 1.

Example 3-50 Enabling zfc

```
echo 1 > /sys/bus/ccw/drivers/zfc/loglevel_erp
```

The log level for zFCP traces has four different levels from 0 to 3. The larger the value becomes, the more detailed is the information that is obtained. The log levels of all areas are set to 0 by default, and only minimum information is output. The debug message is written in the `/var/log/messages` by setting the log level of zfc. To stop the zFCP trace, see Example 3-51.

Example 3-51 Disabling zfc

```
echo 0 > /sys/bus/ccw/drivers/zfc/loglevel_erp
```

Example 3-52 shows the output for the zFCP driver.

Example 3-52 zFCP trace for loglevel_erp

```
Mar  5 10:56:36 linuxk kernel: zfc: zfc_fsf_req_send(4926): calling do_QDIO adapter
0.0.f441, flags=0x2, queue_no=0, index_in_queue=42, count=1, buffers=00000000d5a2260
Mar  5 10:56:36 linuxk kernel: zfc: zfc_fsf_req_send(4933): free_count=127
Mar  5 10:56:36 linuxk kernel: zfc: zfc_fsf_send_fcp_command_task(3757): Send FCP
Command initiated (adapter 0.0.f441, port 0x5005076300cf9b9e, unit
0x5501000000000000)
Mar  5 10:56:36 linuxk kernel: zfc: zfc_qdio_request_handler(310): adapter
0.0.f441, first=42, elements_processed=1
Mar  5 10:56:36 linuxk kernel: zfc: zfc_qdio_zero_sbals(866): zeroing BUFFER 42 at
address 00000000d59fa00
Mar  5 10:56:36 linuxk kernel: zfc: zfc_qdio_request_handler(326): free_count=128
Mar  5 10:56:36 linuxk kernel: zfc: zfc_qdio_request_handler(329):
elements_processed=1, free count=128
Mar  5 10:56:36 linuxk kernel: zfc: zfc_qdio_response_handler(376): first BUFFERE
flags=0x40000000
```

SCSI trace

Using SCSI traces, we would be able to obtain important debug information about the SCSI layer. As already mentioned, it is effective to use with the zFCP trace. Example 3-53 shows the command by which we can set the log level of the SCSI trace.

Example 3-53 Log level setting of SCSI

```
echo 0x9249249 > /proc/sys/dev/scsi/logging_level
```

Example 3-54 shows an example SCSI trace.

Example 3-54 Example SCSI trace

```
Mar 28 22:32:48 T60410A kernel: sd_open: disk=sda
Mar 28 22:32:48 T60410A kernel: scsi_block_when_processing_errors: rtn: 1
Mar 28 22:32:48 T60410A kernel: sd_init_command: disk=sda, block=62, count=8
Mar 28 22:32:48 T60410A kernel: sda : block=62
Mar 28 22:32:48 T60410A kernel: sda : reading 8/8 512 byte blocks.
Mar 28 22:32:48 T60410A kernel: scsi_add_timer: scmd: 000000001e48f900, time: 3000,
(000000002090d740)
Mar 28 22:32:48 T60410A kernel: scsi <0:0:1:0> send 0x000000001e48f900   Read (10) 00
00 00 00 3e 00 00 08 00
Mar 28 22:32:48 T60410A kernel: buffer = 0x000000001ea5e700, buflen = 4096, done =
0x0000000020810a14, queuecommand 0x00000000208c3f1c
```

Usually, the zFCP and SCSI traces are used together to sort out problems occurring in the SCSI and zFCP areas. Use caution when working in the target area and adjusting the log level of the traces. There could be a negative impact to performance when setting higher log levels.

Network problem determination

This chapter discusses network problem determination for Linux on System z. The focus of this chapter is on Linux running on z/VM. We discuss the following topics:

- ▶ Overview of networking options for Linux on System z
- ▶ Network interface detection and configuration under Linux on System z
- ▶ The art of network problem determination
- ▶ Case study

4.1 Overview of networking options for Linux on System z

The networking options available to Linux on System z can be split broadly 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 a System z server. *Virtualization technology* covers the networking options available to those users who plan to run Linux in a z/VM environment only. The z/VM operating system can use any of the physical networking options as well. Thus, Linux systems running as virtual machines (VMs) in a z/VM environment have the choice of using any of the physical options, any of the virtualization technology options, or a combination of both, including:

- ▶ Physical networking options
 - Open Systems Adapters
 - HiperSockets
- ▶ Virtualization technology
 - Guest LAN
 - z/VM virtual switch (VSwitch)

4.1.1 Open Systems Adapters

The Open Systems Adapter (OSA) card is the network adapter for the System z server to connect to the outside world (Open Systems). Depending on the exact type of the OSA card used, 10 Gigabit Ethernet (10 G or 10 GbE), Gigabit Ethernet (GbE), 1000Base-tT Ethernet, Fast Ethernet (FENET), Token-Ring, and Asynchronous Transfer Mode (ATM) are supported. For Linux connectivity, these cards are recognized by the hardware I/O configuration as of the following channel types:

- ▶ Queued Direct I/O (QDIO)
- ▶ Non-Queued Direct I/O (OSE)

Note: For a detailed description of OSA-Express operating modes and limitations, refer the IBM Redbooks publication *OSA-Express Implementation Guide*, SG24-5948.

Queued Direct I/O mode

Queued Direct I/O (QDIO) is a highly efficient data transfer mechanism. It reduces system overhead and improves throughput by using system memory queues and a signaling protocol to directly exchange data between the OSA card microprocessor and the TCP/IP stack.

Non-Queued Direct I/O (non-QDIO) mode

In non-QDIO mode, the data follows a longer I/O path. Linux uses the LCS device driver to communicate with the device when it is running in this mode. Figure 4-1 shows the difference in the QDIO and non-QDIO data paths.

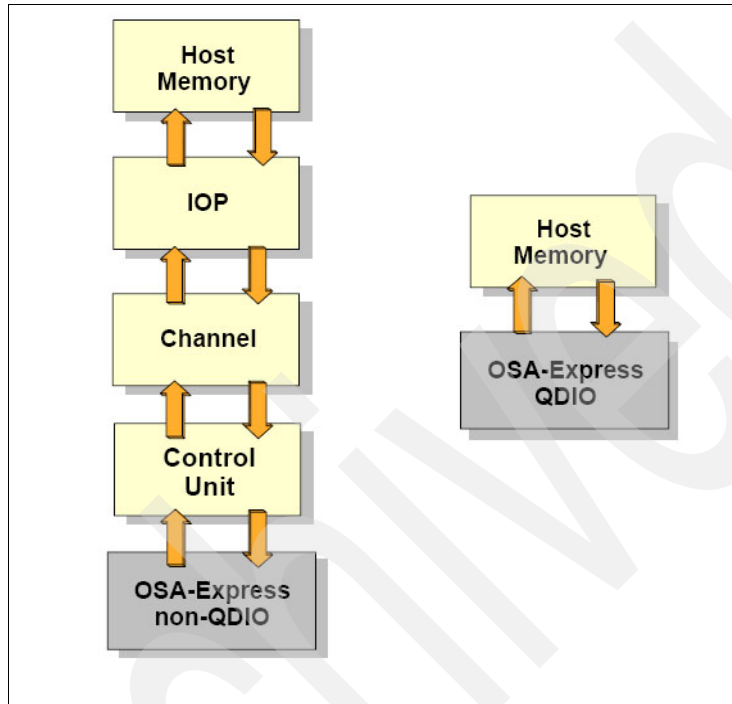


Figure 4-1 QDIO and non-QDIO data paths

Transport modes

For IP and non-IP workloads, the current OSA cards support two transport modes: Layer 2 (Link or Ethernet layer) and Layer 3 (network or IP layer). Each mode processes network data differently.

Layer 2:

- ▶ Uses the MAC destination address to identify hosts and send and receive Ethernet frames.
- ▶ Transports Ethernet frames (not IP datagrams) to and from the operating system TCP/IP stack and the physical network.
- ▶ Does not ARP offload. ARP processing is performed by each operating system TCP/IP stack.
- ▶ Supports MAC level unicast, multicast, and broadcast.

Layer 3 has the following characteristics:

- ▶ Data is transmitted based on the IP address.
- ▶ Only IP addresses are used to identify hosts on the LAN segment.
- ▶ All ARP processing is done by the adapter itself, not by any operating system sharing the adapter.
- ▶ A single MAC address is managed by the adapter for all guests sharing the adapter port.

LPAR-to-LPAR communication

A port of the OSA card can be shared across multiple LPARS. Also, access to a port on the card can be shared concurrently among multiple TCP/IP stacks within the same LPAR. When port sharing, the OSA card running in the QDIO mode has the ability to send and receive IP traffic between LPARs without sending the IP packets over the network.

For outbound packets, the OSA card uses the next hop address provided by the TCP/IP stack to determine where to send the packet. If this next-hop address had been registered by another TCP/IP stack sharing this OSA card, the packet is delivered directly to that TCP/IP stack, and not sent over the LAN. This makes possible the routing of IP packets within the same host system.

Note: Port sharing is supported only between ports that are of the same transport mode, for example, Layer 2 with Layer 2 and Layer 3 with Layer 3.

4.1.2 HiperSockets

HiperSockets provides very fast TCP/IP communications between servers running in different LPARs on a System z machine. Each HiperSocket is defined as a CHPID of type IQD.

To communicate between servers running in the same System z server, HiperSockets sets up I/O queues in the System z server memory. The packets are then transferred at memory speeds between the servers, thereby totally eliminating the I/O subsystem overhead and any external network latency.

HiperSockets implementation is based on the OSA QDIO protocol. Therefore, HiperSockets is called the internal QDIO (iQDIO). HiperSockets is implemented in microcode that emulates the Logical Link Control (LLC) layer of an OSA card.

From a Linux on System z perspective, the HiperSockets interface looks a lot like an OSA QDIO mode interface. Linux uses the qdio and qeth modules to exploit HiperSockets.

4.1.3 Guest LAN

Starting with z/VM V4.2, the z/VM Control Program (control program (CP) has been enhanced to provide a feature known as *Guest LAN*. This feature enables you to create multiple Virtual LAN segments within a z/VM environment. As can be seen from Figure 4-2, Guest LANs do not have a physical connection to the external network. Instead, they must use a router (z/VM TCP/IP or Linux). The Linux router or the z/VM TCP/IP stack must have an external interface such as an OSA card, and an interface connected to the LAN.

Note: Although the structures and the simulated devices related to the Guest LAN under z/VM are virtual, we use the term Guest LAN and not Virtual LAN, because the term Virtual LAN (VLAN) has a different meaning in the networking world.

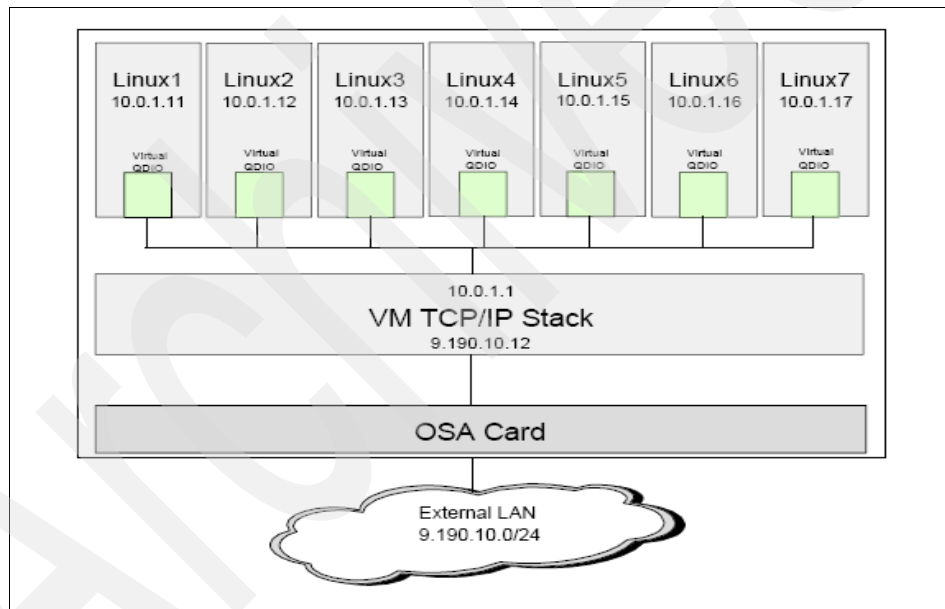


Figure 4-2 z/VM Guest LAN

4.1.4 Virtual Switch

The z/VM Virtual Switch (VSWITCH) introduced with z/VM V4.4 builds on the Guest LAN technology delivered in earlier z/VM releases. VSWITCH connects a Guest LAN to an external interface network using an tOSA card port. Two additional OSA card ports can be specified as backups in the VSWITCH

definition. The Linux guest connected to the VSWITCH is on the same subnet as the OSA card ports 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 the IP packets, and hence can be used only for TCP/IP-related applications. In z/VM V5.1, the VSWITCH implementation was extended to also have the ability to operate at Layer 2 (data link layer) of the OSI model. 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.

4.1.5 Summary of networking options for Linux on System z

Figure 4-3 shows a summary of the networking options that we have discussed for Linux on System z.

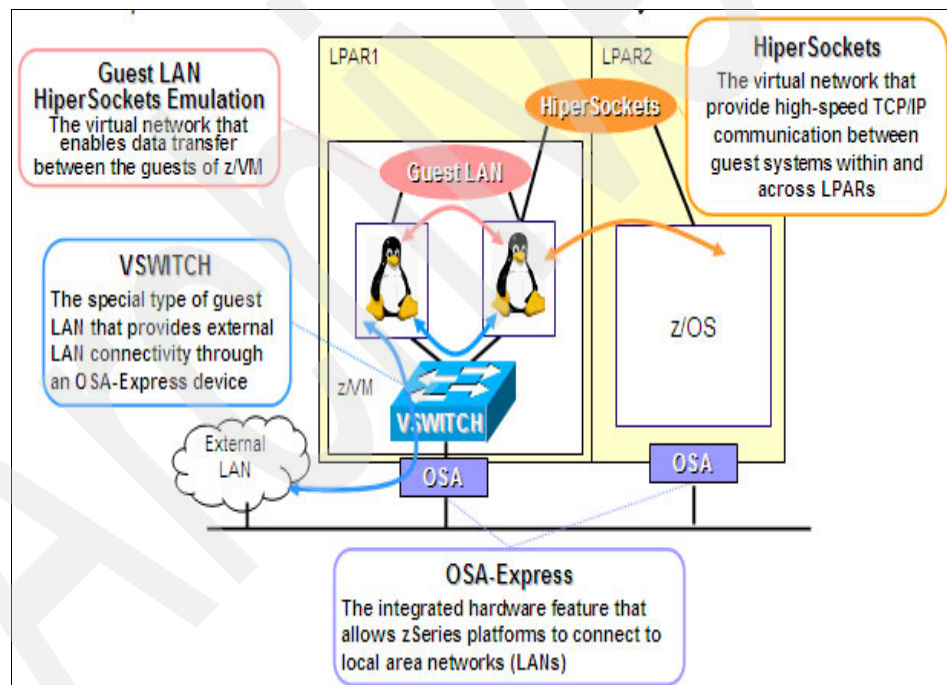


Figure 4-3 Summary of important networking options for Linux on System z

4.2 Network interface detection and configuration under Linux on System z

The network interface detection and configuration under Linux on System z happens through multiple steps. We touch upon some of the important points related to the network interface detection and configuration in this section.

Note: Configuration files and boot or command execution messages from SuSE Linux Enterprise Server are used as examples throughout this chapter. However, the concept applies to Red Hat Enterprise Linux as well.

4.2.1 The Linux hotplug system

The Linux hotplug system is used for managing devices that are connected to the system. Strictly speaking, the Linux hotplug project has two functions within it, called the coldplug and the hotplug. Devices connected at boot time, and some of the not-so-easy-to-detect devices are detected by coldplug. For PCI devices, there is a positive and negative list of device types that should be initialized or skipped by the coldplug, respectively. The hotplug is used to manage devices that can be plugged or unplugged dynamically. The hotplug also handles any device detection task after the kernel is booted.

The hotplug scripts (also known as the *.rc scripts) print a character for each scanned device. Table 4-1 lists these characters with their meanings.

Table 4-1 Characters printed by the Linux hotplug scripts for each scanned device

Character	Meaning
. (period)	The device is already initialized and therefore skipped.
* (asterisk)	Generate a hotplug event for this device.
W	The device is not there in the white list and is therefore skipped.
B	The device is there in the black list and is therefore skipped.

4.2.2 Flow of network interface detection and activation under Linux

Figure 4-4 summarizes the network interface detection for Linux on System z. The hotplug or the coldplug scripts ensure that the correct modules are loaded with the correct options. Then the interface names are allocated. Once the interface names are allocated, appropriate configuration files are read and network parameters are configured.

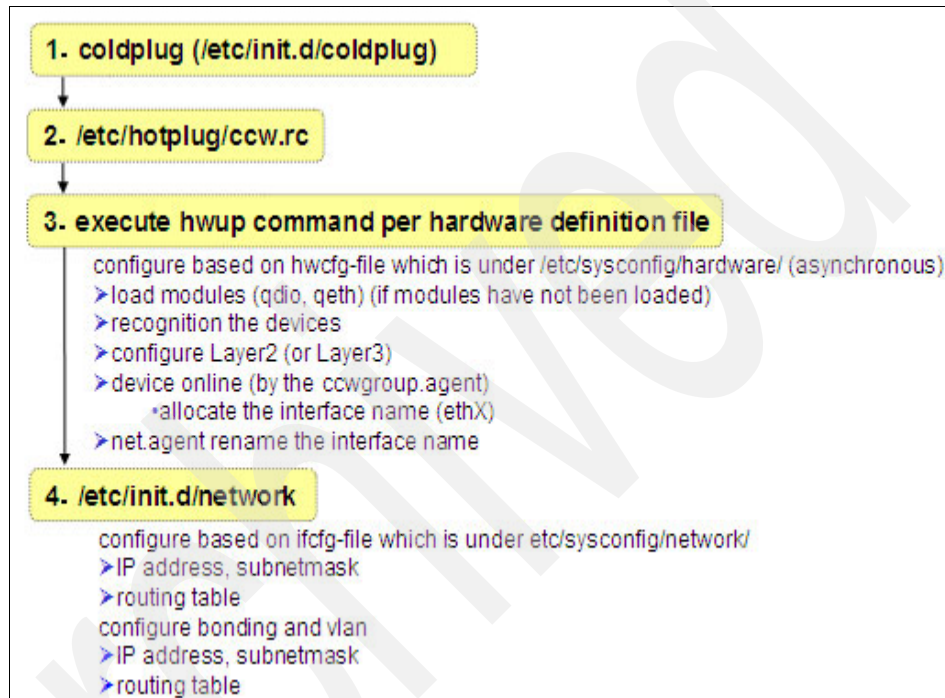


Figure 4-4 Linux network interface detection and activation flow

Figure 4-5 on page 117 shows the sequence in which a network interface is detected with respect to time when the system is coming up. As the diagram shows, the coldplug system calls the **hwup** command to bring the devices up. Usually, there will be three interfaces associated with each network interface, and it is the duty of the ccwgroup.agent to group these devices form an interface and allocate an interface name. Once the interface names are allocated, the network scripts will assign the IP address and other parameters for the interface. Depending on the configuration, the network interface may come up by default, or manual intervention will be required to bring it up.

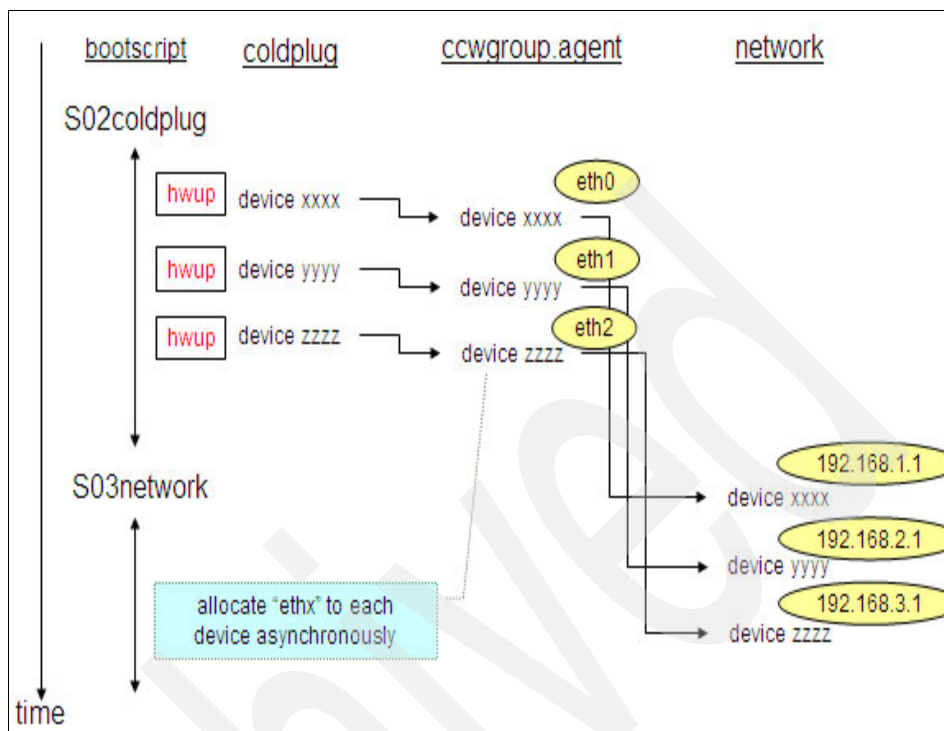
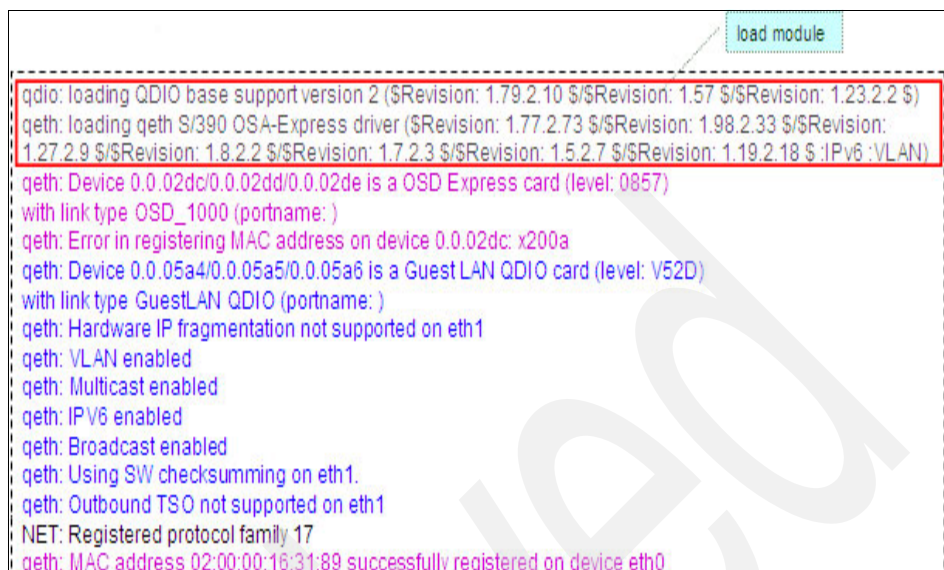


Figure 4-5 Linux network interface detection sequence

Figure 4-6 on page 118 shows sample boot messages generated by a Linux system when the network interface is detected and enabled. The top part of the messages indicate that the **qdio** and **qeth** modules are loaded in to the memory. These modules are required for a Linux on System z system to communicate with the network device in the QDIO mode. The next lines indicate the details of the device detected. In Figure 4-6 on page 118, the fourth line from top indicates that the device addresses are **0.0.02dc**, **0.0.02dd**, and **0.0.02de**. OSD in the same message line indicates that the OSA card is configured in the QDIO mode. Subsequent lines indicate the MAC address, interface names, and list of features enabled on the card.



```
qdio: loading QDIO base support version 2 ($Revision: 1.79.2.10 $/$Revision: 1.57 $/$Revision: 1.23.2.2 $)
qeth: loading qeth S/390 OSA-Express driver ($Revision: 1.77.2.73 $/$Revision: 1.98.2.33 $/$Revision: 1.27.2.9 $/$Revision: 1.8.2.2 $/$Revision: 1.7.2.3 $/$Revision: 1.5.2.7 $/$Revision: 1.19.2.18 $:IPv6:VLAN)
qeth: Device 0.0.02dc/0.0.02dd/0.0.02de is a OSD Express card (level: 0857)
with link type OSD_1000 (portname: )
qeth: Error in registering MAC address on device 0.0.02dc: x200a
qeth: Device 0.0.05a4/0.0.05a5/0.0.05a6 is a Guest LAN QDIO card (level: V52D)
with link type GuestLAN QDIO (portname: )
qeth: Hardware IP fragmentation not supported on eth1
qeth: VLAN enabled
qeth: Multicast enabled
qeth: IPv6 enabled
qeth: Broadcast enabled
qeth: Using SW checksumming on eth1.
qeth: Outbound TSO not supported on eth1
NET: Registered protocol family 17
qeth: MAC address 02:00:00:16:31:89 successfully registered on device eth0
```

Figure 4-6 Boot messages related to network interface detection under Linux on System z

Figure 4-7 indicates the boot messages related to a Layer 3 configuration as well as those for a Layer 2 configuration.

```
layer3
qeth: loading qeth S/390 OSA-Express driver ($Revision: 1.77.2.73 $/$Revision: 1.98.2.33 $/$Revision:
1.27.2.9 $/$Revision: 1.8.2.2 $/$Revision: 1.7.2.3 $/$Revision: 1.5.2.7 $/$Revision: 1.19.2.18 $ :IPv6 :VLAN)
qeth: Device 0.0.05a4/0.0.05a5/0.0.05a6 is a Guest LAN QDIO card (level: V520)
with link type GuestLAN QDIO (portname: )
qeth: Hardware IP fragmentation not supported on eth0
qeth: VLAN enabled
qeth: Multicast enabled
qeth: IPv6 enabled
qeth: Broadcast enabled
qeth: Using SW checksumming on eth0.
qeth: Outbound TSO not supported on eth0

layer2
qdio: loading QDIO base support version 2 ($Revision: 1.79.2.10 $/$Revision: 1.57 $/$Revision: 1.23.2.2 $)
qeth: loading qeth S/390 OSA-Express driver ($Revision: 1.77.2.73 $/$Revision: 1.98.2.33 $/$Revision:
1.27.2.9 $/$Revision: 1.8.2.2 $/$Revision: 1.7.2.3 $/$Revision: 1.5.2.7 $/$Revision: 1.19.2.18 $ :IPv6 :VLAN)
qeth: Device 0.0.02dc/0.0.02dd/0.0.02de is a OSD Express card (level: 0857)
with link type OSD_1000 (portname: )
qeth: Error in registering MAC address on device 0.0.02dc: x200a
NET: Registered protocol family 17
qeth: MAC address 02:00:00:16:31:89 successfully registered on device eth0
```

Figure 4-7 Boot messages related to network Layer 3 and Layer 2 configurations

4.2.3 Network interfaces and their interface names

On systems with multiple network interfaces, the interface name allocated to the OSA devices may not be consistent across Linux system reboots. The reason for this is that the hotplug events are not synchronous, and multiple drivers initialize their devices in parallel. This inconsistency in the interface name may lead to trouble for certain middleware such as Oracle®. Also, SNMP and the Linux monitoring tool sar may also be affected by this issue.

Figure 4-8 illustrates the asynchronous nature of the hotplug events. As illustrated, the device 3333 is configured ahead of device 222 and now has the interface name of eth1 allocated to it instead of the expected eth2. Device 222 is the eth2 now.

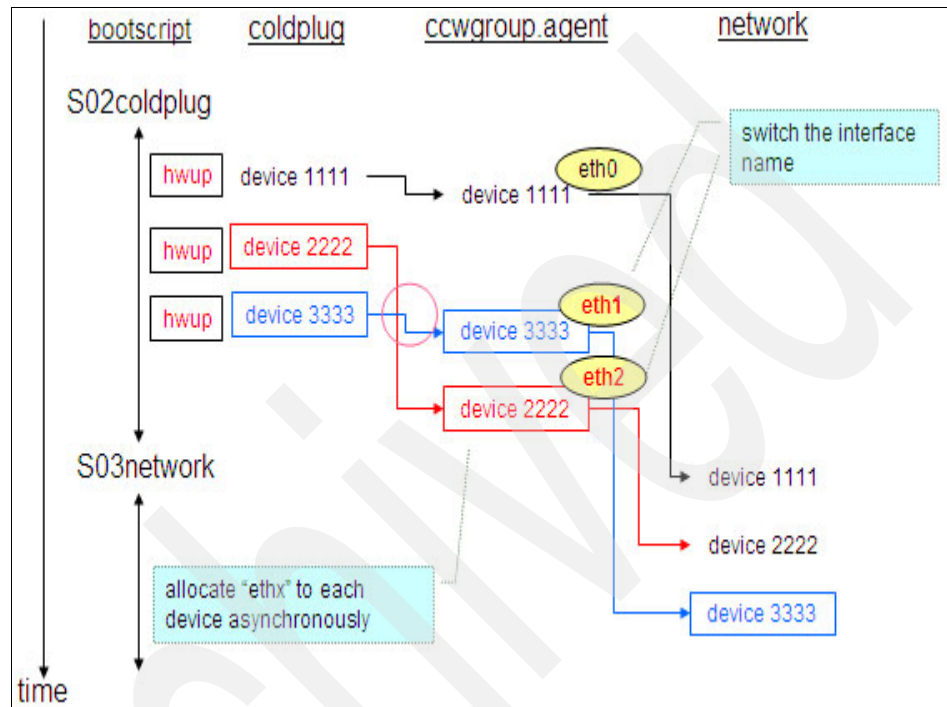


Figure 4-8 Asynchronous hotplug events and inconsistent interface naming

To avoid this problem, the hotplugging of the network interfaces can be queued by editing the `/etc/sysconfig/hotplug` and adding the entry:

```
HOTPLUG_PCI_QUEUE_NIC_EVENTS=yes
```

Another solution is to use **udev** and make the connection between the device address and the interface name in the file `/etc/udev/rule.d/30-net_persistent_names.rules`. Example 4-1 shows how eth6 can be permanently assigned to 0.0.05a2.

Example 4-1 Allocating the interface name eth6 to the device 0.0.05a2 permanently

```
SUBSYSTEM=="net", ACTION=="add", ENV{PHYSDEVPATH}=="*0.0.05a2",
IMPORT="/lib/udev/rename_netiface %k eth6"
```


When channel bonding is used, this problem will not occur since channel bonding does not use hotplug. Hence, the device names remain the same across reboots of the Linux system.

4.2.4 Network configuration files overview

In this section we take a quick look at some of the important configuration files for network devices on a Linux system running on System z. As noted earlier, SuSE Linux Enterprise Server files are used as examples.

/etc/sysconfig/hardware/hwcfg-qeth-ccw-0.0.xxxxfile

This file is referred to by the **hwup** and **hwdown** commands. Figure 4-9 shows a sample **hwcfg-qeth-ccw-0.0.xxxx** file. For each OSA device on the system, there will be a unique **hwcfg-qeth** file. The device in this case is 0.0.02d4, and hence the exact file name is **hwcfg-qeth-ccw-0.0.02d4**.

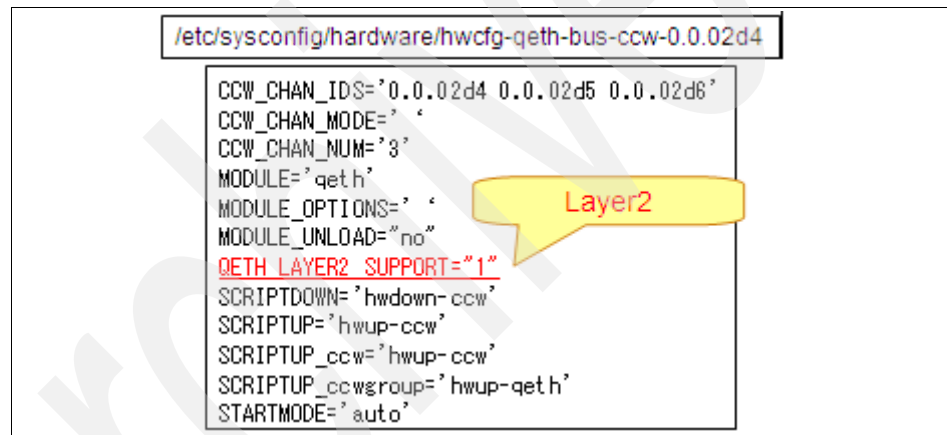


Figure 4-9 Network device is defined in **/etc/sysconfig/hardware/hwcfg-qeth-ccw-xx**

/etc/sysconfig/network/ifcfg-qeth-ccw-0.0.xxxx file

This configuration file is used by the **ifup** and **ifdown** commands. Figure 4-10 on page 122 shows a sample **ifcfg-qeth-ccw-0.0.xxxx** file. For each OSA device on the system, there will be a unique **ifcfg-qeth** file. The device in this case is 0.0.02dc, and hence the exact file name is **ifcfg-qeth-bus-ccw-0.0.02dc**. The IP

address, subnet mask, local MAC address, MTU, automatic startup (STARTMODE), and so on, are configured in this file. When STARTMODE=onboot, the interface is activated automatically during the system startup.

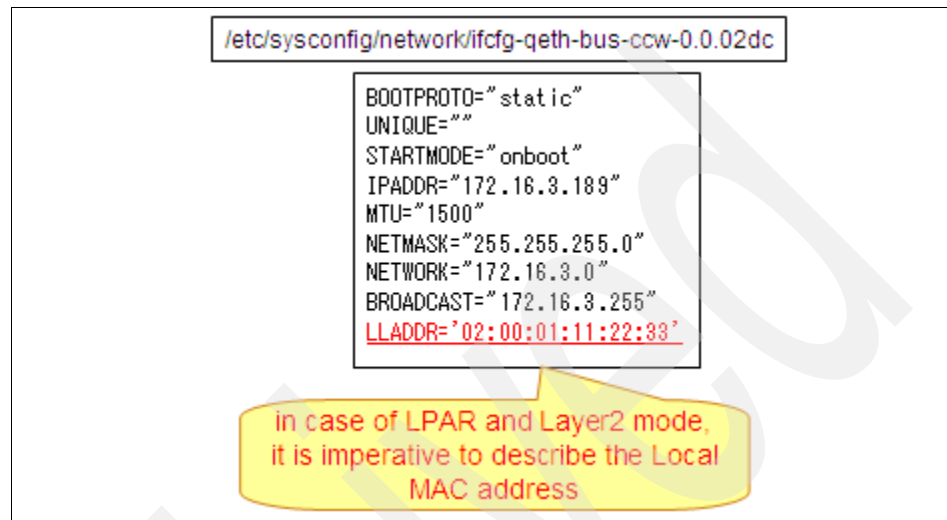


Figure 4-10 IP address is configured in `/etc/sysconfig/network/ifcfg-qeth-bus-ccw-xxx`

`/etc/sysconfig/network/routes` file

This file is used by the `ifup` and `ifdown` commands. This file is used for configuring static network routes. Figure 4-11 shows a sample network routes file. The format of the file is destination network (192.168.30.0 in this case), next-hop (172.16.3.131 in this case), subnetmask (255.255.255.0), and the interface name (qeth-bus-ccw-0.0.05a4) in this case. The second line shown in Figure 4-11 is a special entry that defines the default gateway. The keyword *default* identifies it as the default gateway. The two minus signs (- -) indicate that they assume the default values.

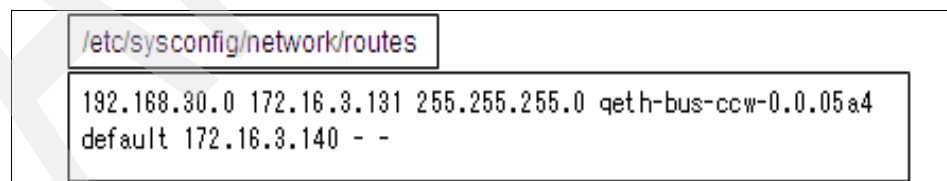


Figure 4-11 Network routes are configured in `/etc/sysconfig/network/routes`

4.3 Network problem determination

Having quickly covered the common networking options for Linux on System z, the network interface detection and configuration under Linux, in this section we discuss network problem determination. The previous two sections should act as a foundation for this section.

The network problems can be broadly classified into two categories:

- ▶ Problems related to network communication
- ▶ Problems related to network performance

Problems related to the network communication are discussed in detail in this chapter. Chapter 5, “Performance problem determination” on page 163, is dedicated to performance problem determination on Linux on System z and discusses network performance problems as well.

4.3.1 Problems related to network communication

Communication involves more than one party. Such is the case of computer systems. When computer systems exchange data or messages themselves, we say that they are communicating. When this exchange of messages breaks, we say that there is a communication problem. So, at a very high level, a network communication problem on a system can be a problem local to that system, or it can be outside of that system (that is, the local machine network is functioning correctly). In our discussion, we assume that the local system is a Linux system running on the System z server, and from that system we are trying to communicate to other systems.

Problem determination on the local machine

When a system is not able to communicate with other systems, there are some logical questions that we need to ask. Are you able to communicate to *any* system on the network? Did we make *any* changes to the system prior to the problem occurrence? Are we able to communicate to some of the systems? Is there any information related to the problem in the error logs? We will look at each of these shortly.

Network communication layer for Linux on System z

Figure 4-12 on page 124 shows the network communication layer for Linux on System z. Such a layer is very useful to narrow down the problem. The bottom-most layer is the hardware (HW) layer. The OSA device falls into this layer. The next layer is the virtual machine (VM) layer, and the virtualization done at the z/VM level (such as the VSWITCH) falls into this layer. The next layers, QDIO and QETH, represent the device driver. The MAC address is represented

by the ARP layer. Of course, the IP address gets mapped to the IP layer, and all TCP/UDP-related components to the TCP/UDP layer. Finally, applications such as SSH, FTP, Telnet, and so on, are represented by the application layer.

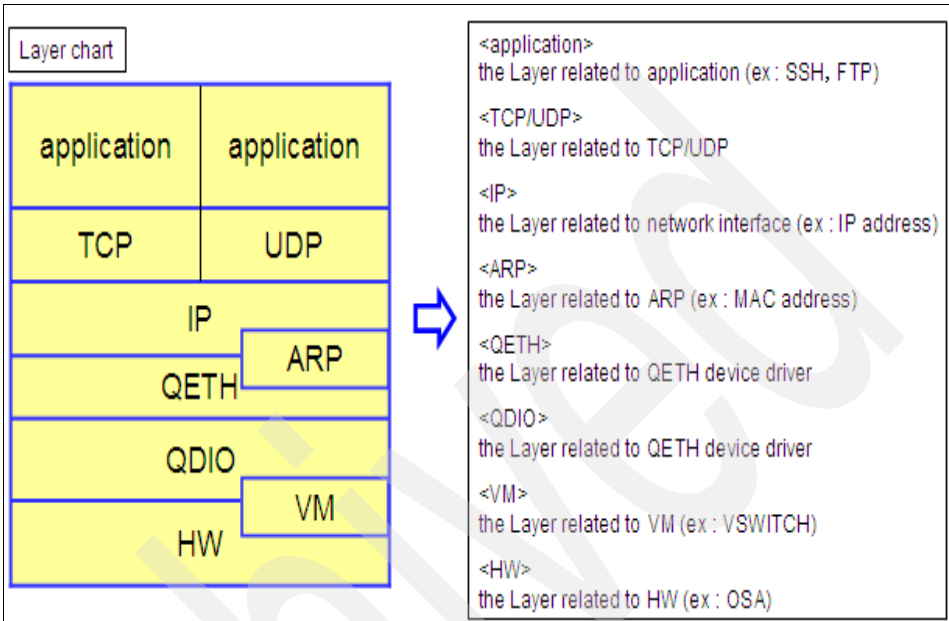


Figure 4-12 Network communication layer for Linux on System z

Figure 4-13 compares the network communication layer for Linux on System z with the OSI reference model. The hardware layer in the OSI reference model is made up of the HW, VM, QDIO, and QETH layers. The data link layer of the OSI model is represented by the ARP layer. Also, part of the ARP layer, along with the IP layer, makes up the network layer of OSI. The OSI transport layer is implemented as the TCP/UDP layer and, finally, the application layer in the network communication model for Linux on System z represents the session, presentation, and application layers of the OSI reference model.

Application	Application
Presentation	
Session	
Transport	TCP•UDP
Network	IP•ARP
Data Link	ARP
Physical	HW•VM•QETH•QDIO

Figure 4-13 Network communication layer for Linux on System z and the OSI reference model

Problem determination flow chart

While trying to narrow down a problem, it is important to analyze any information already available. Figure 4-14 on page 126 shows a high-level flow chart demonstrating how this can be done. This flow chart describes an approach that can be used for problem determination on the local system. Usually, when there is a problem, the first thing to try is to capture as many error logs as possible. In some cases it may happen that the error messages get overwritten. So, it is very important to try capturing the error messages as and when they are available. An examination of error messages will help us to narrow down the problem to a layer or at least to a set of layers. When a working system suddenly stops working, before jumping in and editing configuration files, it is important to investigate whether any changes were made on the system before the problem occurred. In most cases we will notice that there was a change made and that it resulted in an error. If the problem is occurring on a newly installed system, it is worth checking all the configurations for any mistake. Sometimes changes made at the network

level (outside the System z) server can also create problems for the systems running on the System z server. So, good coordination with the network support staff is also an important factor in troubleshooting network problems. In some cases, it may be required to use network packet tracing tools to narrow down or isolate the problem. Usually, then there is a permanent communication issue, and it is easy to identify and fix compared to the case of an intermittent problem. Intermittent problems may take more time to troubleshoot and may need to be kept under observation for some time after fixing to ensure that it is really fixed.

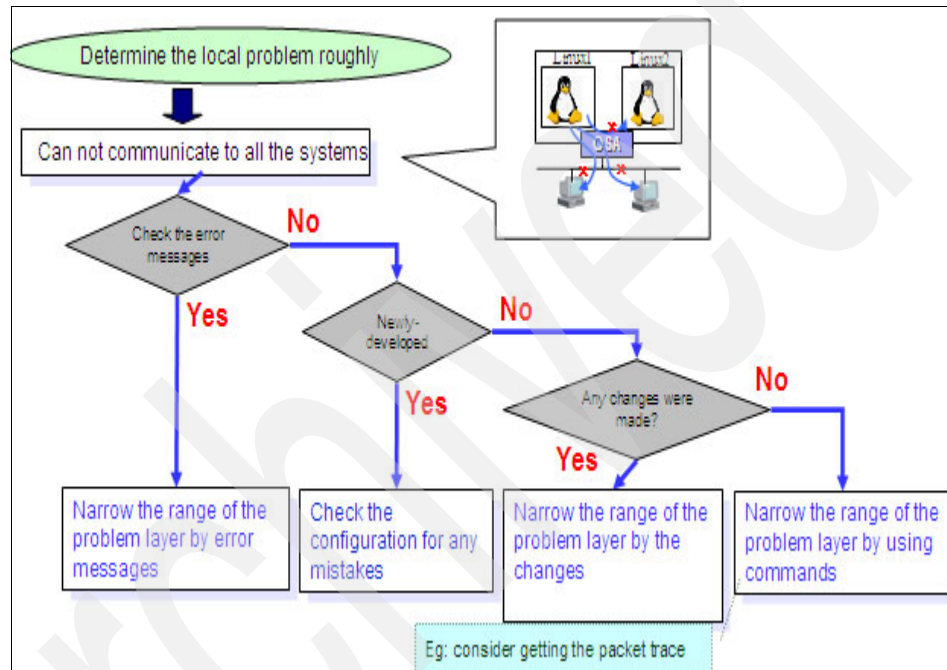


Figure 4-14 Problem determination on the local system

In certain cases it is possible that there is not much useful information available to analyze for problem determination. In such cases, it is better to execute specific commands and analyze their output for problem determination. Figure 4-15 shows a simple but common and powerful way of using the **ping** command to narrow down the problem with respect to some of the layers. As can be seen from the diagram, if the result of a **ping** command is a success, then we are sure that the network cable is connected, the network interface is active, and IP communication (network layer) is happening. In such a case, it is certain that the problem is somewhere above the IP layer.

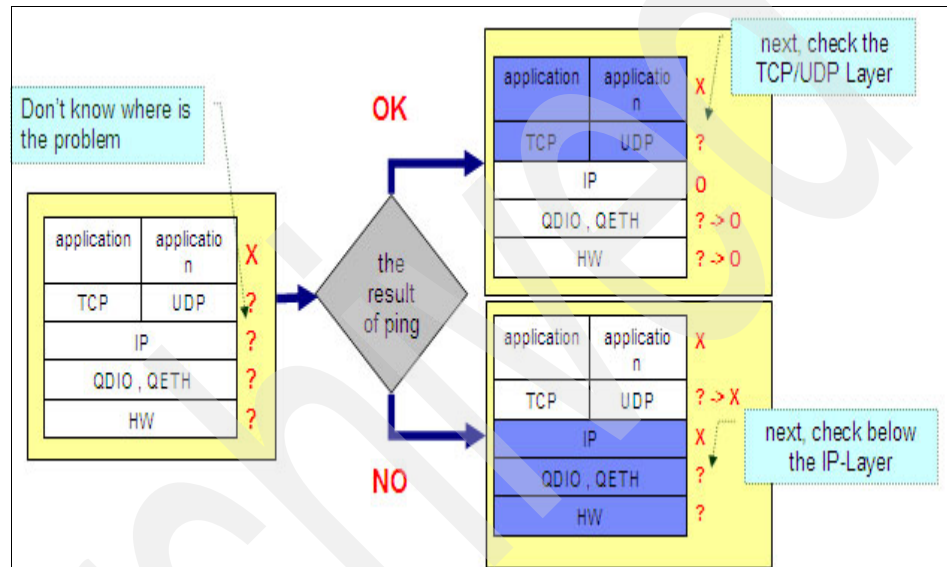


Figure 4-15 Using the ping command to narrow down to the problem layers

On the other hand, if the ping was a failure, then it is clear that there is some problem below the TCP/UDP layer itself. This could be a hardware problem such as the cable not being connected, and so on. In this case, it is clear that the first level troubleshooting should start at the IP layer level and below.

We now discuss how to survey in each layer to narrow down a problem.

Survey in the IP layer

Because of the simplicity of the **ping** command, it is usually better to start at the IP layer and narrow up or down while we troubleshoot a problem.

Figure 4-16 summarizes the common tasks one could perform to survey in the IP layer.

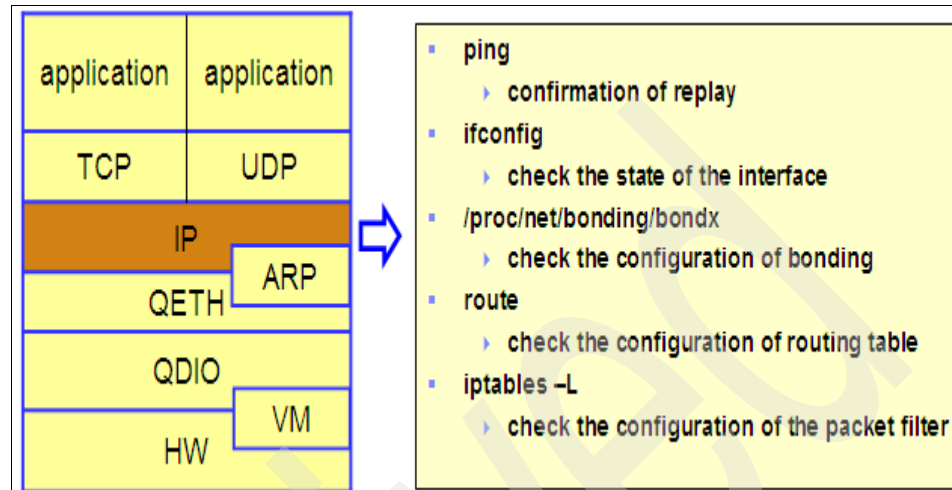


Figure 4-16 Survey in the IP layer

Figure 4-17 shows a sample ping output. The ping command tells us whether we could reach the destination address and, if yes, how much time it took. So, if there was no reply from the destination address (meaning, not able to reach), then we should continue our check below the IP layer. If there was a reply, then we should continue our check above the IP layer.

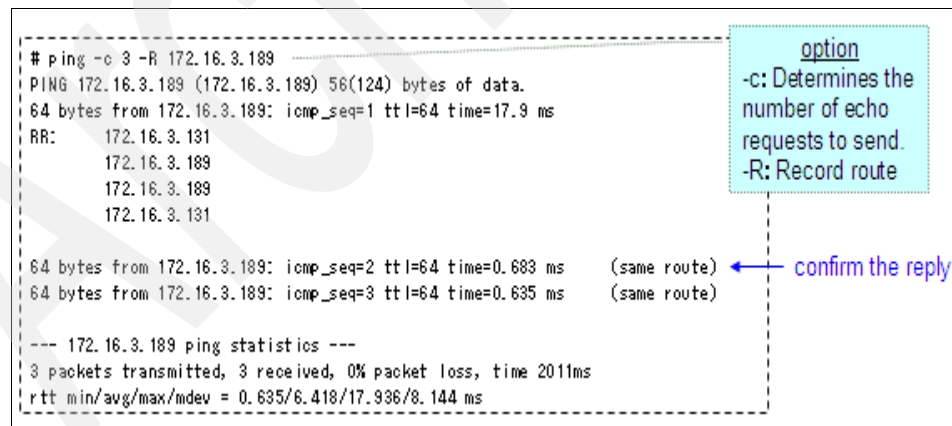


Figure 4-17 Using ping for diagnosis

Figure 4-18 shows the output of another useful command, **ifconfig**. Important things that we need to check in the output are indicated in the diagram.

```
# ifconfig
bond0    Link encap:Ethernet  HWaddr 02:00:00:16:31:89 (1)
         inet addr:172.20.3.189 (2) Bcast:172.20.3.255  Mask:255.255.255.0 (3)
         inet6 addr: fe80::ff:fe16:3189/64 Scope:Link
         UP (4) BROADCAST RUNNING MASTER MULTICAST  MTU:1500 (5) Metric:1
         RX packets:437 (6) errors:0 dropped:0 overruns:0 frame:0
         TX packets:542 (7) errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:36252 (35.4 Kb)  TX bytes:47336 (46.2 Kb)

...

(1)MAC address      (2)IP address
(3)subnetmask      (4)state of the interface
(5)MTU              (6)the number of received packets
(7)the number of send packets
```

Figure 4-18 Using **ifconfig** for diagnosis

When the **ifconfig** command is executed without any command-line parameters, only the interfaces that are active (that is, UP) are displayed. The **-a** command-line parameter will force **ifconfig** to display all the interfaces that are configured irrespective of their state. Figure 4-19 shows a sample **ifconfig** command output when executed with the **-a** command-line parameter.

```
# ifconfig -a
eth0     Link encap:Ethernet  HWaddr 00:11:25:C0:1A:00
         BROADCAST MULTICAST  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0

eth1     Link encap:Ethernet  HWaddr 02:00:00:00:00:2A
...

It shows that eth0 is DOWN, because of not displaying the "UP"
set the MAC address of OSA port
```

Figure 4-19 **ifconfig** command output indicating interface DOWN state

If the interface is down, then the troubleshooting should continue below the IP layer. If the IP address, subnet mask, and so on, are not displayed correctly, then the respective configuration file (/etc/sysconfig/network/ifcfg-qeth-ccw-0.0.xxxx

for SLES) should be checked. In case of layer-2 configuration, the local MAC address will also be configured in this file.

The **route** command, when executed with the -n option, displays the kernel routing table (**netstat -rn** will also show a similar output). Figure 4-20 shows a sample **route -n** output.

```
# route -n
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
172.20.3.0	*	255.255.255.0	U	0	0	0	vlan10
172.16.3.0	*	255.255.255.0	U	0	0	0	eth1
192.168.30.0	172.16.3.131	255.255.255.0	UG	0	0	0	eth1
169.254.0.0	*	255.255.0.0	U	0	0	0	eth1
127.0.0.0	*	255.0.0.0	U	0	0	0	lo
0.0.0.0	172.16.3.140	0.0.0.0	UG	0	0	0	eth1

Figure 4-20 Checking the routing table

Sometimes the communication is not happening due to a packet filtering setup. The packet filtering (firewall) configuration can be checked using the iptables command. Figure 4-21 shows a sample iptables -L output that indicates that SSH connections coming from any host should be dropped.

```
# iptables -L
iptables -L
Chain INPUT (policy ACCEPT)
target    prot opt source                destination
DROP      tcp  --  anywhere              anywhere           tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target    prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target    prot opt source                destination
```

DROP the connection by ssh

Figure 4-21 Checking the packet filtering configuration by using iptables

Survey in the ARP layer

Figure 4-22 summarizes the ARP layer survey commands.

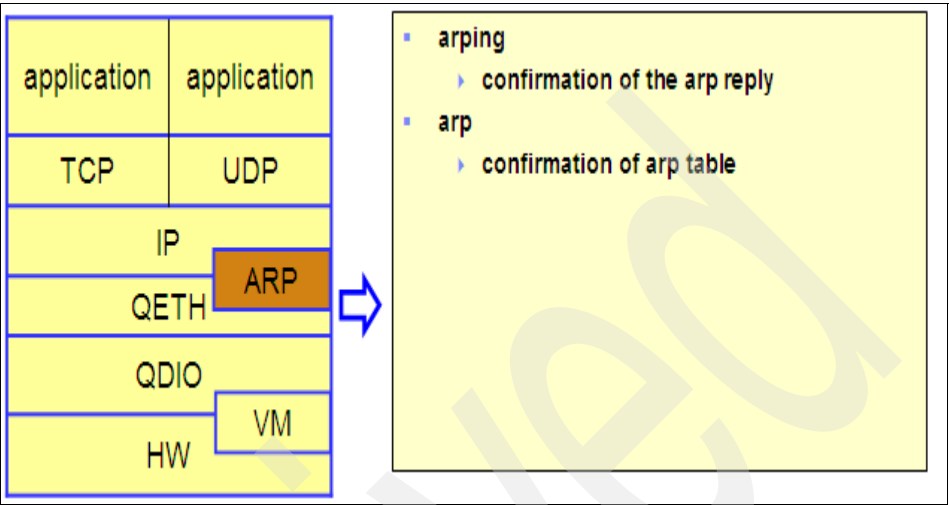


Figure 4-22 Survey in the ARP layer

The **arping** command can be used to send ARP requests to a host. Figure 4-23 shows a sample output of an **arping** command execution. Such an ARP request updates the ARP cache of the host on the same network.

```
# arping -I bond0 -c 3 172.20.3.182
ARPING 172.20.3.182 from 172.20.3.189 bond0
Unicast reply from 172.20.3.182 [00:03:47:98:77:6A] 0.990ms
Unicast reply from 172.20.3.182 [00:03:47:98:77:6A] 0.957ms
Unicast reply from 172.20.3.182 [00:03:47:98:77:6A] 0.922ms
Sent 3 probes (1 broadcast(s))
Received 3 response(s)
```

Figure 4-23 Using arping for diagnosis

If there was no response, the troubleshooting should continue below the ARP layer. Figure 4-24 shows how to check the ARP table using the `arp` command.

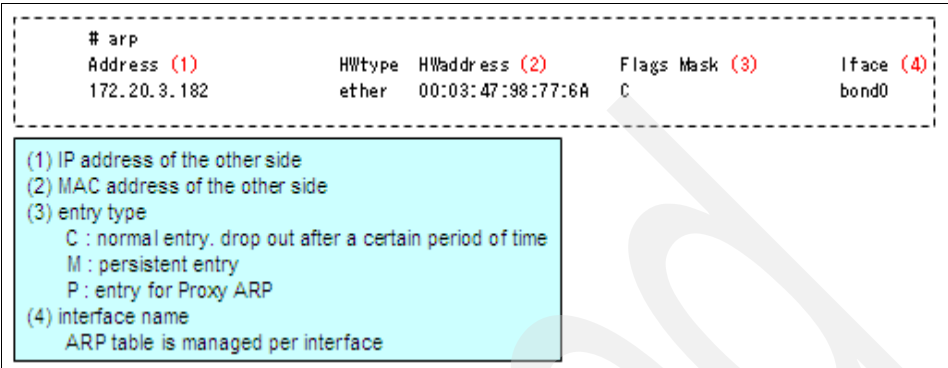


Figure 4-24 Using the `arp` command for diagnosis

Survey in the QETH layer

Figure 4-25 shows the different things that can be surveyed in the QETH layer. The main thing that needs to be checked in this layer is whether the qeth device driver has recognized the devices. Figure 4-26 on page 133 shows a sample output of the `/proc/qeth` file indicating how to check whether the network device is recognized by the qeth device driver.

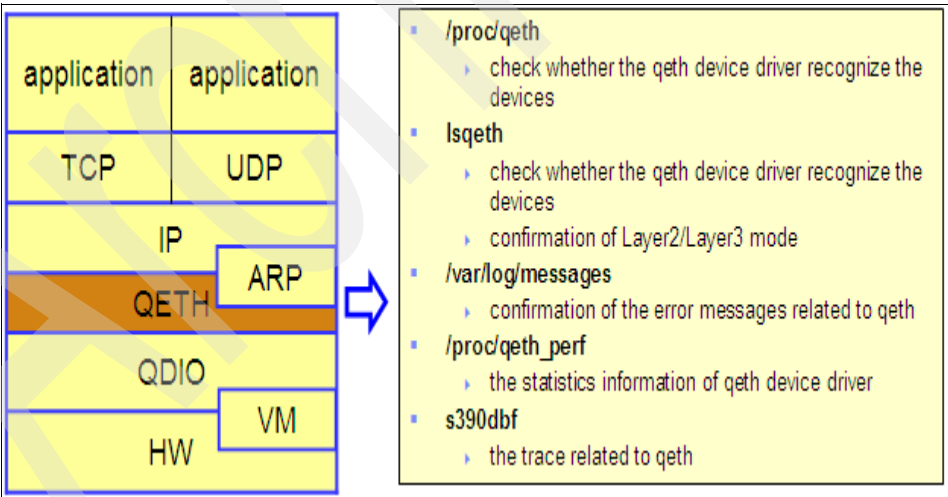


Figure 4-25 Survey in the QETH layer

If the device is not recognized, then check the device configuration in /etc/sysconfig/hardware/hwcfg-xxx.

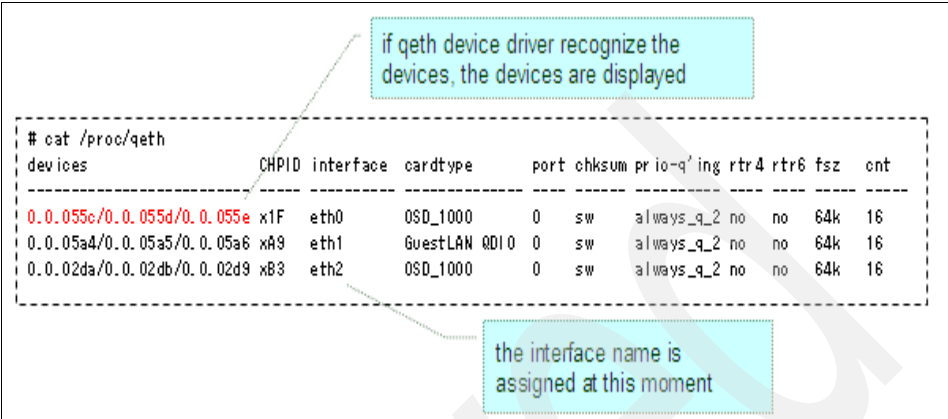


Figure 4-26 Using the /proc/qeth file information for diagnosis

Figure 4-27 shows the output of the **lsqeth** command, which is another useful command output to understand whether the device is online and in which layer (Layer 2 or Layer 3) it is operating. If the output is not as expected, then the device configuration in the /etc/sysconfig/hardware/hwcfg-xxx file should be checked first.

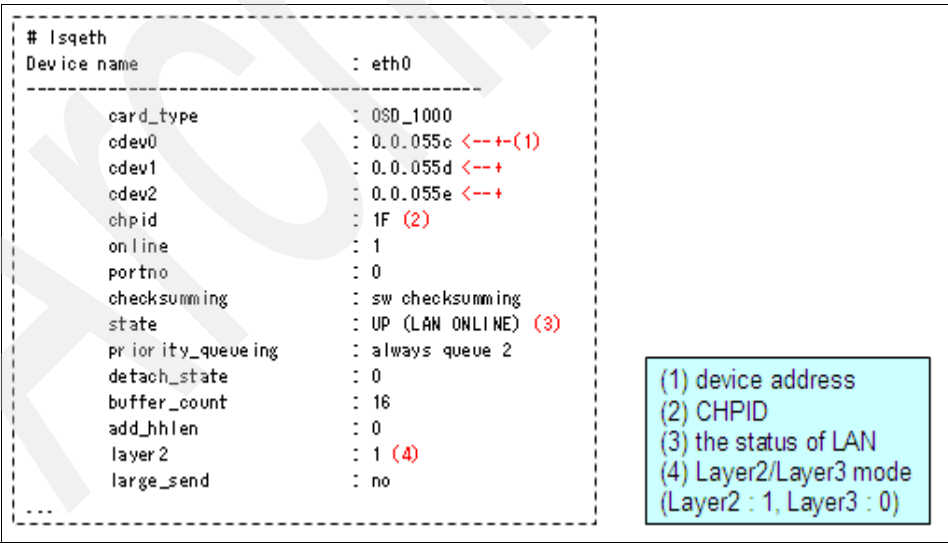


Figure 4-27 Using the lsqeth output for diagnosis

The `/var/log/messages` file in Linux captures most of the common messages generated. This is a very important file referenced by system administrators, developers, and so on, to troubleshoot a problem. Figure 4-28 shows sample messages logged on to the `/var/log/messages` file while there was a problem with one of the network devices. The device name coming at the top of the sample output shown helps us to determine whether the problem is with the device that we are working with.

```
qeth: check on device 0.0.3600, dstat=x0, cstat=x4 <4>
qeth: irb: 00 c2 40 17 03 77 50 38 00 04 00 00 00 00 00 00
qeth: irb: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
qeth: irb: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
qeth: irb: 00 00 00 00 00 00 00 00 00 00 00 02 00 15 00 5d
qdio : received check condition on activate queues on device 0.0.3602 (cs=x4, ds=x0).
qeth: Recovery of device 0.0.3600 started ...
bonding: bond1: link status definitely down for interface eth2, disabling it
bonding: bond1: making interface eth3 the new active one.
```

Figure 4-28 Using the `/var/log/messages` information for diagnosis

The number of packets transferred between the `qdio` and `qeth` drivers can be checked by examining the `/proc/qeth_perf` file. Figure 4-29 shows a sample output with the number of packets transferred between the `qeth` and `qdio` indicated. Unexpectedly large packet counts mean that there is a problem at the `qeth` driver level or at the hardware level.

```
# cat /proc/qeth_perf
For card with devnos 0.0.0240/0.0.0241/0.0.0242 (eth0):
Skb's/buffers received      : 16144384/11280405
Skb's/buffers sent         : 3694744/3694744
. . .
Inbound do_QDIO time (in us) : 1114771
Inbound do_QDIO count       : 1410050
Outbound handler time (in us) : 4472000
Outbound handler count      : 3694743
Outbound time (in us, incl QDIO) : 45266773
Outbound count              : 3694744
Outbound do_QDIO time (in us) : 41639121
Outbound do_QDIO count      : 3694744
```

the number of packets from QDIO to QETH

the number of packets from QETH to QDIO

Figure 4-29 Using information from `/proc/qeth_perf` for diagnosis

Additional trace information for debugging can be found in the /proc/s390dbf/qeth* files. While reporting a problem to IBM support, the content of these files needs to be provided for diagnosis. Figure 4-30 shows the output of trace information for qeth.

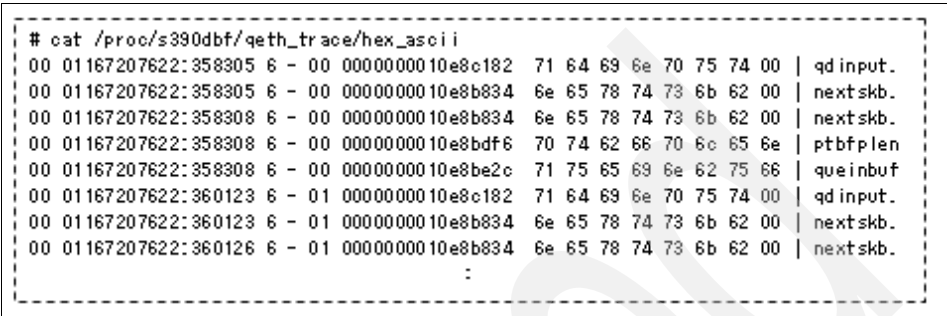


Figure 4-30 Using information from /proc/s390dbf/qeth_trace/hex_ascii for diagnosis

Survey in the QDIO layer

The survey in the QDIO layer is essentially checking the trace information from the debugger. Figure 4-32 on page 136 shows a sample output of the /proc/s390dbf/qdio_trace/hex_ascii.

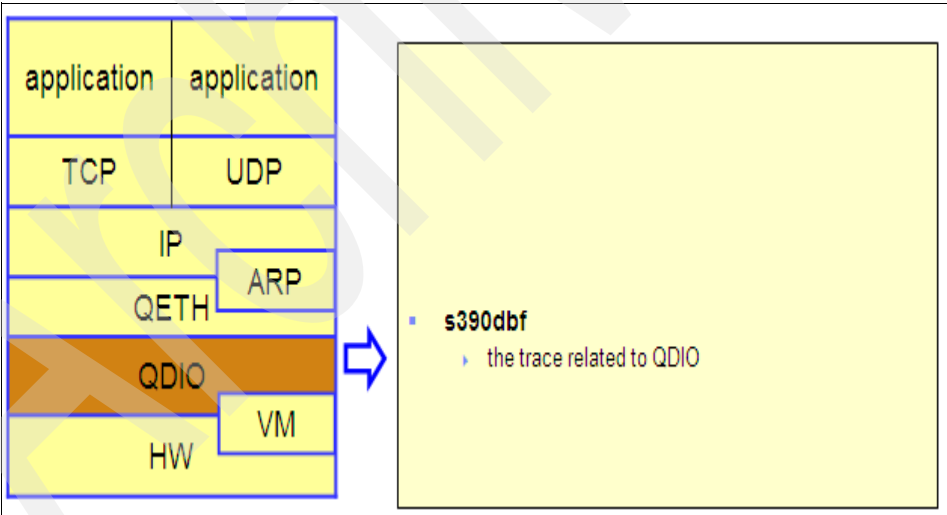


Figure 4-31 Survey in the QDIO layer

```
# cat /proc/s390dbf/qdio_trace/hex_ascii
00 01177658169:814289 0 - 01 000000001082097a 71 61 6c 63 20 20 20 38 | qalc 8
00 01177658169:814368 0 - 01 000000001082097a 71 61 6c 63 20 20 20 37 | qalc 7
00 01177658169:994630 2 - 01 00000000108253ca 64 65 6c 79 74 72 67 74 | delytrgt
00 01177658169:994631 2 - 01 000000001082539a 00 00 00 00 00 00 00 00 | .....
00 01177658169:994632 0 - 01 00000000108250d4 71 65 73 74 20 20 20 37 | qest 7
00 01177658169:996463 0 - 01 0000000010822f4a 71 65 68 69 20 20 20 37 | qehi 7
00 01177658169:996508 2 - 01 0000000010824938 71 61 63 74 20 20 20 37 | qact 7
```

Figure 4-32 Using the information from /proc/s390dbf/qdio_trace/hex_ascii for diagnosis

Survey in the VM layer

Figure 4-33 shows a summary of the useful commands for a survey in the VM layer. While troubleshooting at the VM level, it is important to understand when you are in the virtual world and when you are in the real world.

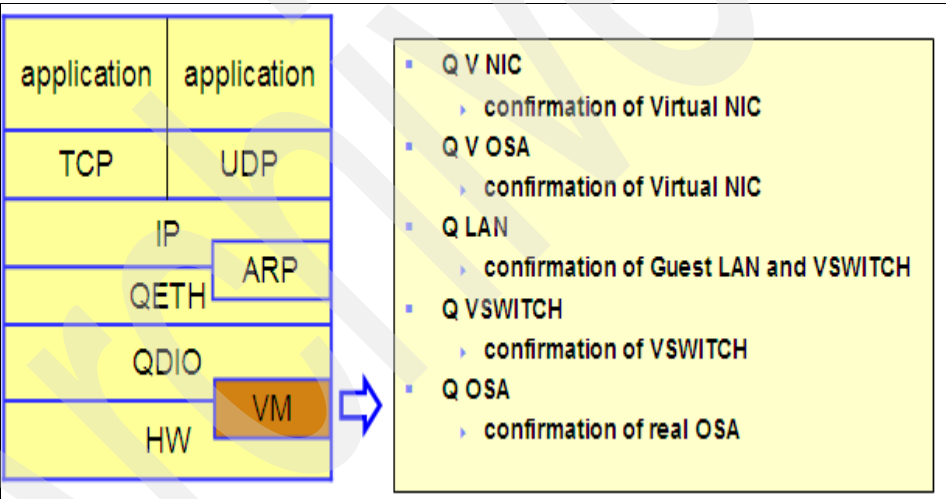


Figure 4-33 Survey in the VM layer

A sample output of the Q V NIC command is shown in Figure 4-34. This command output shows the status of the virtual NIC assigned to the VM user. If the virtual NIC numbers are not correct or if the type of the device is incorrect, then check the z/VM directory (USER DIRECT).

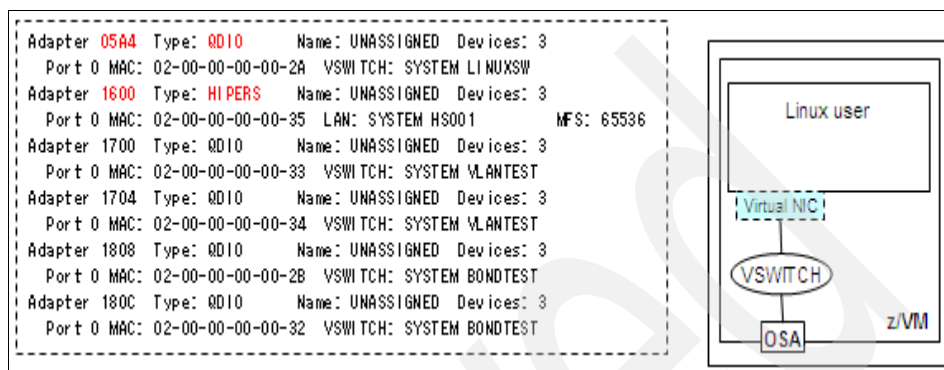


Figure 4-34 Using the Q V NIC output for diagnosis

Figure 4-35 shows the Q V OSA output. This command output also shows the direct attached OSA, which the Q V NIC command will not show. If the correct OSA address is not displayed, then the USER DIRECT file needs to be checked.

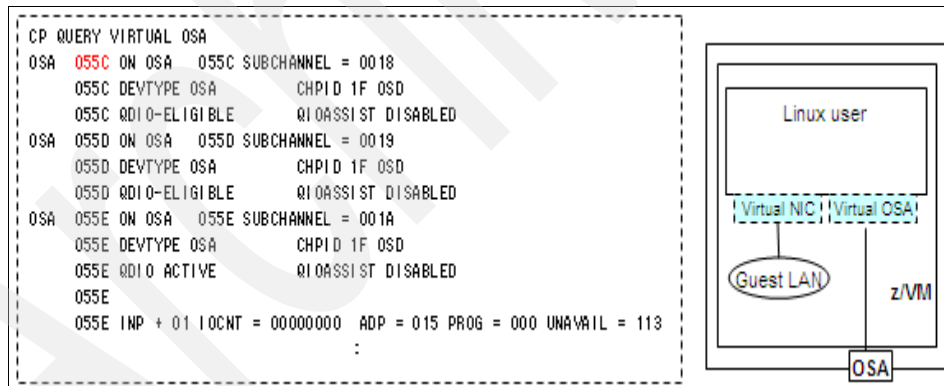


Figure 4-35 Using the Q V OSA output for diagnosis

Figure 4-36 shows the Q LAN output. This command output shows the information related to the HiperSockets, Guest LAN, VSWITCH, and so on. If the LAN information is not displayed correctly, then IOCP needs to be checked in the case of HiperSockets and the z/VM SYSTEM CONFIG file for issues related to the Guest LAN and the VSWITCH.

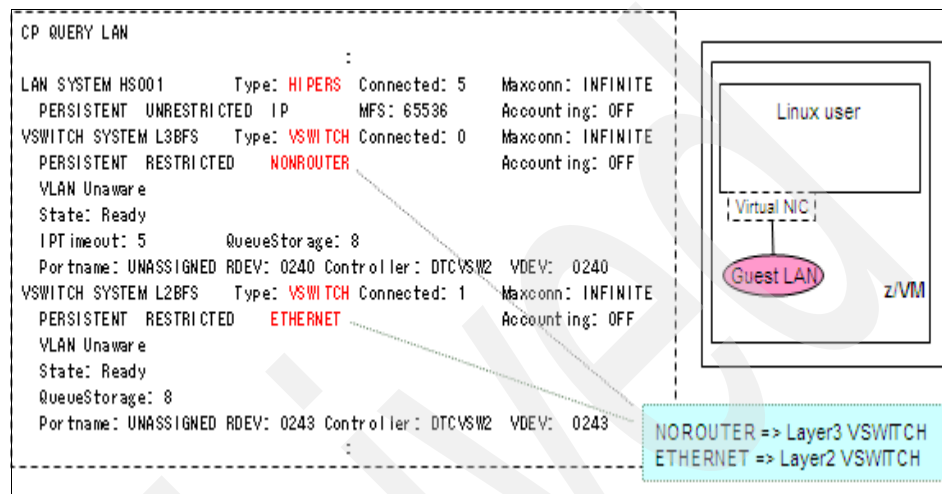


Figure 4-36 Using the CP Q LAN output for diagnosis

The Q LAN ACCESSLIST command output shown in Figure 4-37 helps to show the authority to access the LAN. The authority is defined in the SYSTEM CONFIG file.

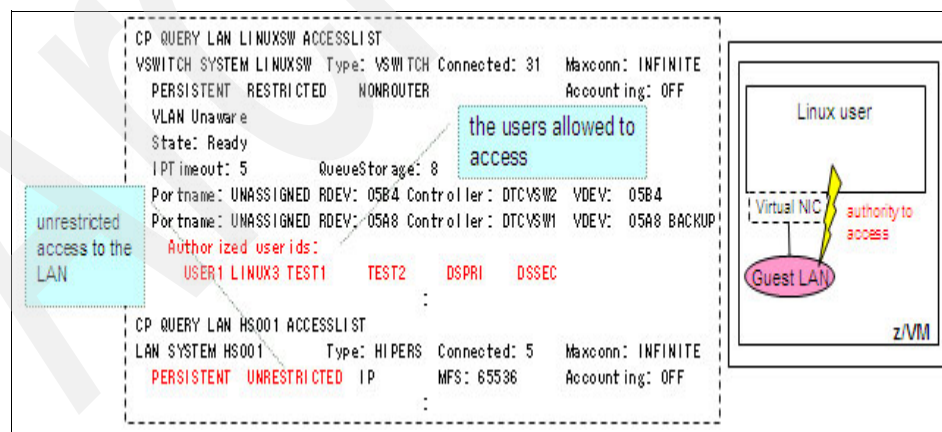


Figure 4-37 Using the Q LAN ACCESSLIST output for diagnosis

The CP Q LAN DETAIL command output, as shown in Figure 4-38, provides information such as the number of packets received and transmitted on the LAN, IP address, and MAC address of the guest systems connected to the LAN, and so on. If the number of packets received and transmitted increases rapidly, then it may be indicating a problem.

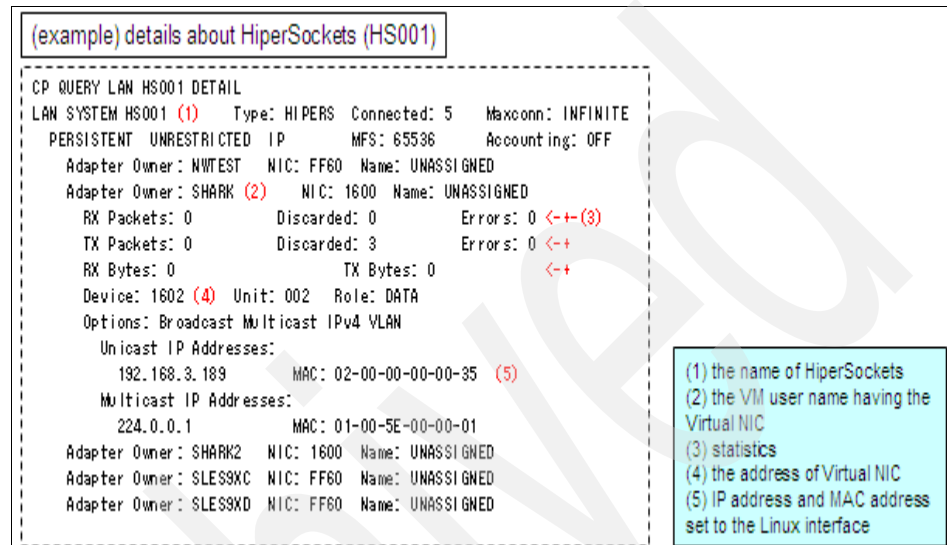


Figure 4-38 Using the CP Q LAN DETAIL output for diagnosis

Similar to the Q LAN command, the Q VSWITCH command with its options can be used to query about the VSWITCH configuration in effect, information about the guest systems connected, packet transmitted and received, and so on.

The Q OSA command output will show information about the real OSA card. If the expected devices are not displayed, then the IOCP needs to be checked. To get more information about OSA, such as the OSA Address Table (OAT), the z/VM OSA/SF tool can be used. Figure 4-39 shows a sample OSA/SF output. This output can be used to verify the MAC addresses and the IP addresses registered.

Image 0.A (CEC1LIN) CULA 0									
00(0550)* MPC	N/A	HALL0LE	(QDIO control)			SIU	ALL		
02(0552) MPC	00 No4	No6 HALL0LE	(QDIO data)			SIU	ALL		
		020101112233	(VMAC)						
		333300000001	(Group MAC)						
03(0553)	N/A					N/A	CSS		
04(0554)* MPC	N/A	z/VM0004	(QDIO control)			SIU	ALL		
06(0556) MPC	00 No4	No6 z/VM0004	(QDIO data)			SIU	ALL		
		010.119.254.065	(REG)						
		010.119.254.066	(REG)						
		:							

Figure 4-39 Using the z/VM OSA/SF tool for diagnosis

Note: For information about OSA/SF, visit:
<http://www.vm.ibm.com/related/osasf>

Survey in the hardware (HW) layer

Figure 4-40 on page 141 summarizes the main points of interest for a survey in the hardware layer. Before touching up on these points, it will be helpful to explain a useful Linux kernel parameter. Usually, when a Linux system is coming up, it senses all the hardware devices connected. Also, Linux senses any dynamic change, such as adding a new device or removing a device when it is running. All the detected devices (irrespective of whether they were detected during the system boot or after that) are controlled under sysfs. Most of these devices can be made online or offline dynamically when Linux is running. It is possible to disable this sensing for a given device by using the cio_ignore kernel parameter. When the cio_ignore parameter is set for a device, that device is not sensed during a system reboot, and cannot be brought online or offline from Linux.

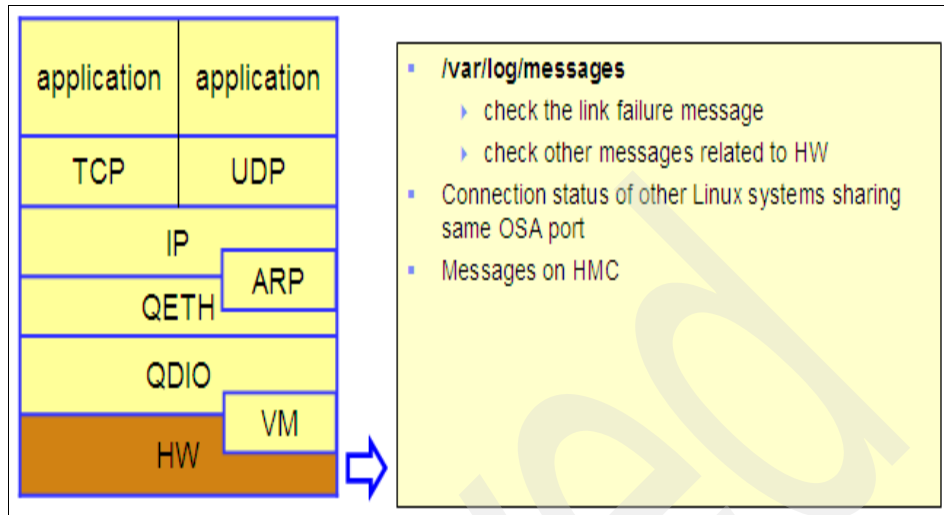


Figure 4-40 Survey in the hardware layer

`cio_ignore` can be set up permanently by adding an entry similar to the one shown in Example 4-2 in to the `/etc/zipl.conf` file. It is also possible to set up the `cio_ignore` for a device dynamically, as shown in Example 4-3.

Example 4-2 Sample entry to set up `cio_ignore` in `/etc/zipl.conf`

```
cio_ignore=0.0.1000-0.0.1fff,0.0.3000-0.0.3fff
```

The devices 1000~1FFFF,3000~3FFF are not sensed.

Example 4-3 Dynamic `cio_ignore` set up with `/proc/cio_ignore`

```
# echo add 0.0.1000 > /proc/cio_ignore
  (the device 1000 is added to cio_ignore list)
# echo free 0.0.2000 > /proc/cio_ignore
  (the device 2000 is removed from cio_ignore list)
```

Figure 4-41 shows the output of the `lscss` command. This output is useful in understanding the channel path information learned by Linux. If the device address provided by the `lscss` command is not correct, then check IOCP, z/VM USER DIRECT, and `cio_ignore`. The `lscss` command does not show a device that is disabled (or hidden) using the `cio_ignore` setting.

# lscss									
Device	Subchan.	DevType	CU	Type	Use	PIM	PAM	POM	CHPIDs
:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:
0.0.05B0	0.0.0008	1732/01	1731/01	yes	80	80	FF	A9000000	00000000
0.0.05B1	0.0.0009	1732/01	1731/01	yes	80	80	FF	A9000000	00000000
0.0.05B2	0.0.000A	1732/01	1731/01	yes	80	80	FF	A9000000	00000000

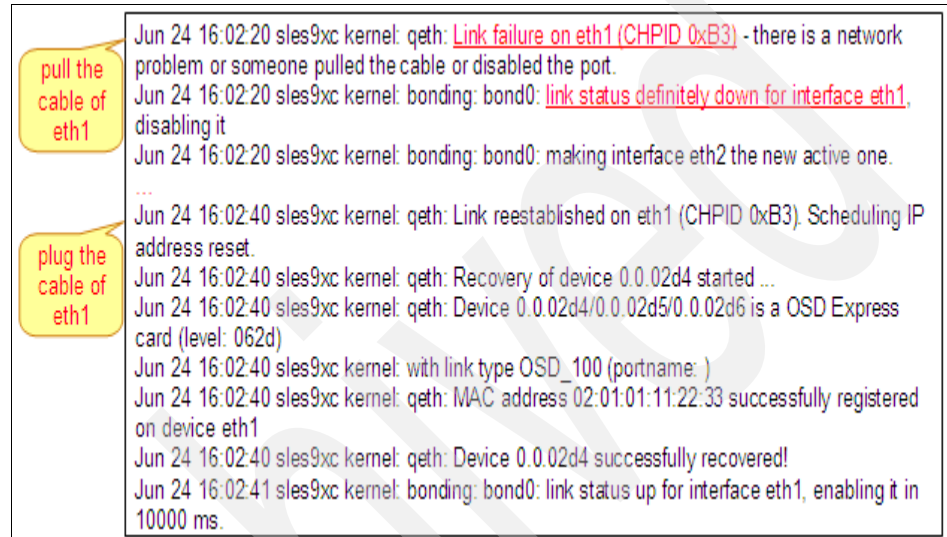
↑
device address

Figure 4-41 Using the `lscss` output for diagnosis

Example 4-4 Sample `/var/log/messages` entry indicating a link failure

Jun 24 16:02:20 sles9xc kernel: qeth: Link failure on eth1 (CHPID 0xB3)
- there is a network problem or someone pulled the cable or disabled the port.

The `/var/log/messages` file will contain information about the link status. Example 4-4 on page 142 shows a sample message in the `/var/log/messages` file indicating link failure for the `eth1` interface. In case of a link failure message, some of the things to check are the cable or the connection to the switch, the status of the other Linux systems sharing the same OSA port, if any, and the hardware messages and return codes displayed on the Hardware Management Console (Hardware Management Console (HMC)).



The screenshot shows a terminal window with the contents of the `/var/log/messages` file. Two yellow callout boxes with red text provide instructions: 'pull the cable of eth1' points to the first log entry, and 'plug the cable of eth1' points to the subsequent log entries. The log entries show a link failure on `eth1` at 16:02:20, followed by a recovery process at 16:02:40, and the link status returning to up at 16:02:41.

```
Jun 24 16:02:20 sles9xc kernel: qeth: Link failure on eth1 (CHPID 0xB3) - there is a network
problem or someone pulled the cable or disabled the port.
Jun 24 16:02:20 sles9xc kernel: bonding: bond0: link status definitely down for interface eth1,
disabling it
Jun 24 16:02:20 sles9xc kernel: bonding: bond0: making interface eth2 the new active one.
....
Jun 24 16:02:40 sles9xc kernel: qeth: Link reestablished on eth1 (CHPID 0xB3). Scheduling IP
address reset.
Jun 24 16:02:40 sles9xc kernel: qeth: Recovery of device 0.0.02d4 started ...
Jun 24 16:02:40 sles9xc kernel: qeth: Device 0.0.02d4/0.0.02d5/0.0.02d6 is a OSD Express
card (level: 062d)
Jun 24 16:02:40 sles9xc kernel: with link type OSD_100 (portname: )
Jun 24 16:02:40 sles9xc kernel: qeth: MAC address 02:01:01:11:22:33 successfully registered
on device eth1
Jun 24 16:02:40 sles9xc kernel: qeth: Device 0.0.02d4 successfully recovered!
Jun 24 16:02:41 sles9xc kernel: bonding: bond0: link status up for interface eth1, enabling it in
10000 ms.
```

Figure 4-42 `/var/log/messages` output indicating link DOWN and UP status

Survey in the TCP/UDP layer

To survey in the TCP/UDP layer, one of the best methods is to use the **netstat** command to check the status of the TCP or UDP socket.

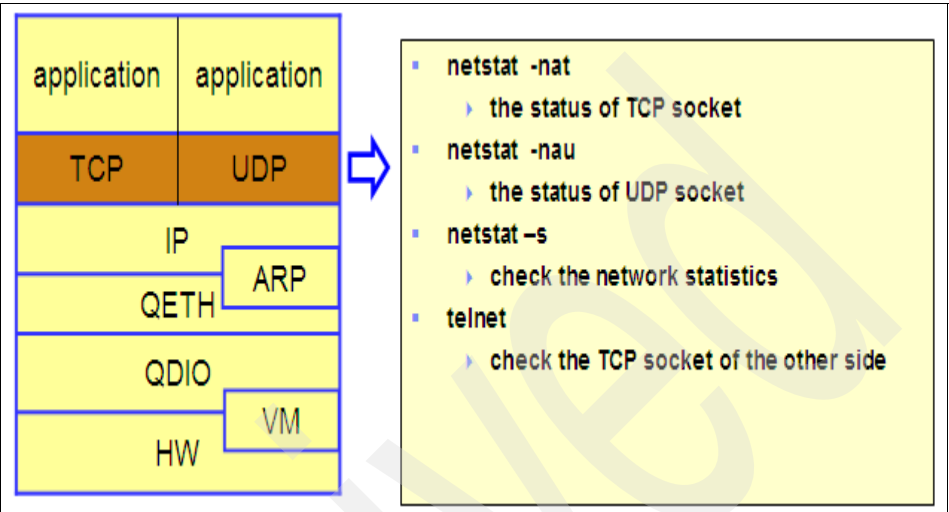


Figure 4-43 Survey in the TCP/UDP layer

Figure 4-44 and Figure 4-45 show sample outputs for TCP (that is, with the -t option) and for UDP (that is, with the -u option) sockets, respectively, using the **netstat** command. From this output, we can find the port on which the local machine is communicating and the destination port, and so on. Usually, when we try to connect to an application, that application will be listening on a port. So, the **netstat** output should be examined to check whether the application of interest is really listening on the port, whether a connection is established, and so on.

the TCP socket information of local node

current status

```
# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN
tcp        0 1452 9.170.20.67:22          9.188.22.138:1413       ESTABLISHED
tcp        0      0 172.16.1.42:22           172.16.3.182:37320      ESTABLISHED
tcp        0      0 172.16.1.42:37827        172.16.3.127:23         ESTABLISHED
```

Figure 4-44 Using the **netstat -nat** output for TCP socket diagnosis

the UDP socket information of own node

```
# netstat -nau
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 0.0.0.0:795             0.0.0.0:*
udp        0      0 0.0.0.0:798             0.0.0.0:*
udp        0      0 0.0.0.0:799             0.0.0.0:*
udp        0      0 0.0.0.0:800             0.0.0.0:*
udp        0      0 0.0.0.0:932             0.0.0.0:*
udp        0      0 0.0.0.0:812             0.0.0.0:*
udp        0      0 0.0.0.0:177             0.0.0.0:*
udp        0      0 0.0.0.0:111             0.0.0.0:*
```

Figure 4-45 Using the **netstat -nau** output for UDP socket diagnosis

Note: The State column will be blank for the netstat -tau output because UDP is a stateless protocol.

Another useful netstat output is shown in Example 4-5. If the count for the segments retransmitted is increasing, then usually it is a point of worry.

Example 4-5 Sample netstat -s output

```
# netstat -s
...
  Tcp:
  1638 active connections openings
  4889 passive connection openings
  31 failed connection attempts
  474 connection resets received
  14 connections established
  8310735 segments received
  7790702 segments send out
  1860 segments retransmitted
  0 bad segments received.
  1275 resets sent
...
```

A very quick way to check whether a system is listening on a TCP port is by doing a Telnet to that system onto that port. This method cannot be used for UDP since it is a stateless protocol. Figure 4-46 on page 147 shows an example

of using the Telnet command to establish a connection to TCP port 111. The results of a successful connection and an unsuccessful connection are displayed.

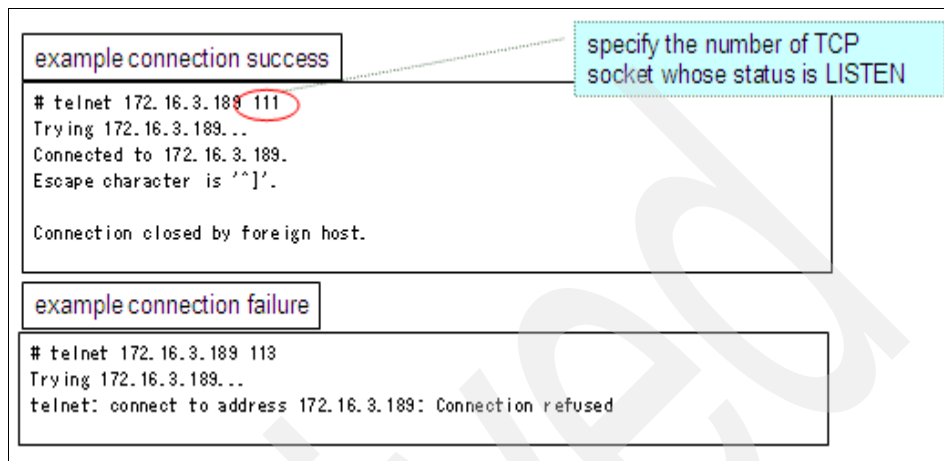


Figure 4-46 Using telnet to diagnose TCP socket status

Note: Keep in mind that when attempting a connection from one system to another on a specific TCP port, all the network devices on the path must allow a connection on that port. Otherwise, even if the socket is open on the destination system, the connection may fail.

Survey in the application layer

The application layer survey can be done by examining whether the required processes are running on the system, by checking the number of open file descriptors, and so on.

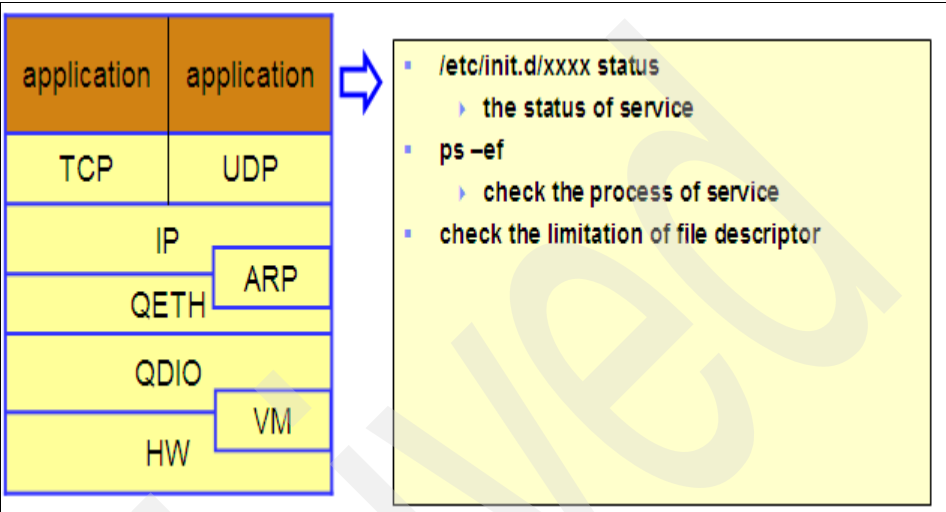


Figure 4-47 Survey in the application layer

Example 4-6 shows a method to check the status of the sshd service. In this case, the output indicates that the SSH daemon is running.

Example 4-6 Checking the status of the SSH server service

```
# /etc/init.d/sshd status
Checking for service sshd                                     running
```

The sample output (last 3 lines) shown in Example 4-7 indicates that the mail service (master and qmgr) is running and the SSH daemon is running with one user connected.

Example 4-7 Sample ps -ef output (modified for simplicity) showing processes

```
# ps -ef
UID      PID  PPID  C  STIME TTY          TIME CMD
root      1    0  0  Apr26 ?        00:00:00 init [5]
root      2    1  0  Apr26 ?        00:00:00 [migration/0]
root      3    1  0  Apr26 ?        00:00:12 [ksoftirqd/0]
root      4    1  0  Apr26 ?        00:00:00 [migration/1]
root      5    1  0  Apr26 ?        00:00:42 [ksoftirqd/1]
root      6    1  0  Apr26 ?        00:00:00 [events/0]
```

```

root      7      1  0 Apr26 ?      00:00:00 [events/1]
root     25233      1  0 Jun07 ?      00:00:00 /usr/lib/postfix/master
postfix  25244 25233  0 Jun07 ?      00:00:00 qmgr -l -t fifo -u
root     26889  1534  0 11:20 ?      00:00:00 sshd: root@pts/2

```

Linux has a feature (ulimit) to limit the number of files opened by each process. This limit is essentially the number of file descriptors each process can use. Linux has a feature (ulimit) that limits the number of files opened by each process. This limit is essentially the number of file descriptors that each process can use. The file descriptor is needed when Linux uses resources such as standard I/O and communication between processes, because these resources are treated virtually as files. Example 4-8 shows the number of file descriptors opened by a process with process ID 4610.

Example 4-8 Sample output showing the number of file descriptors used by a process

```

# ls -l /proc/4610/fd
total 7
dr-x----- 2 root root  0 May 30 20:04 .
dr-xr-xr-x  3 root root  0 May 30 20:04 ..
lrwx----- 1 root root 64 May 30 20:04 0 -> /dev/pts/4
lrwx----- 1 root root 64 May 30 20:04 1 -> /dev/pts/4
lrwx----- 1 root root 64 May 30 20:04 2 -> /dev/pts/4
lrwx----- 1 root root 64 May 30 20:04 3 -> socket:[732707]
lrwx----- 1 root root 64 May 30 20:04 4 -> /dev/pts/4
lrwx----- 1 root root 64 May 30 20:04 5 -> /dev/pts/4
lrwx----- 1 root root 64 May 30 20:04 6 -> /dev/pts/4

```

For network applications such as an http daemon, the number of clients is limited by the file descriptor limitation. So, when new clients are unable to connect to an already running HTTP daemon, it could be that the HTTP daemon has reached the limit where it can no longer open a new file descriptor.

Problem determination outside the local machine

When we are sure that the local machine network is working correctly, then it is necessary to focus the problem determination task externally from the local system. Again, here too it is necessary to ask some logical questions to be able to proceed. Is the communication problem with a specific system only? Is the communication problem with all the other systems? Were there any changes made on the network or on any of the systems? A simple flow chart on this regard is shown in Figure 4-48.

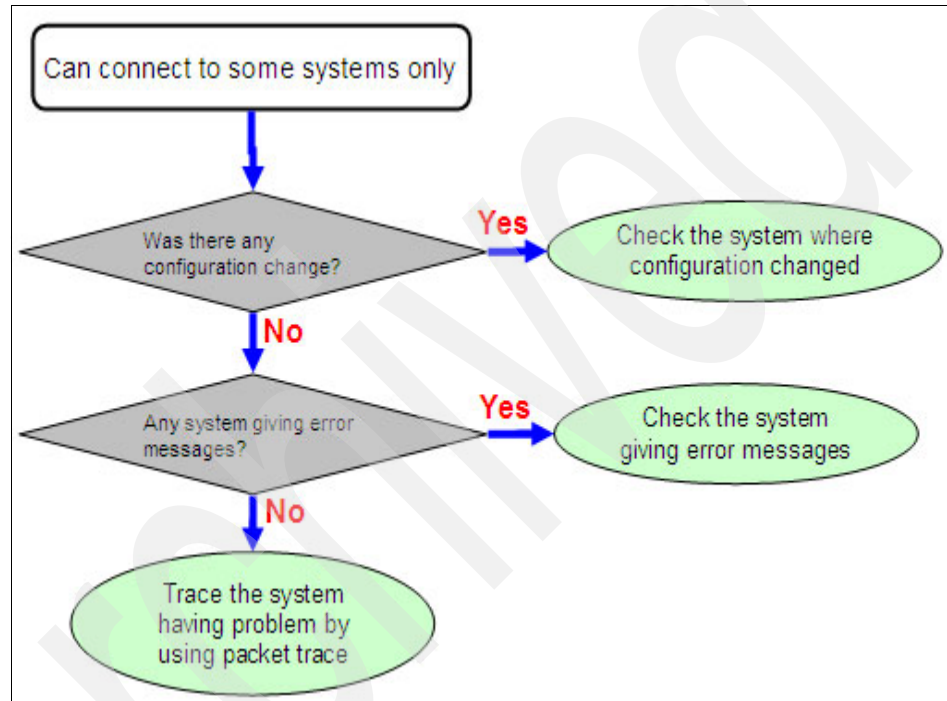


Figure 4-48 Problem determination outside the local machine

Packet tracing

Packet tracing is a very powerful technique for network problem determination. Packet tracing helps us to determine the problem by monitoring the data on the network. It is common to packet trace on the local machine, on the destination machine, and on a network device such as the switch or router. In some cases it may be required to do packet tracing at more intermediate network levels depending on the complexity of the network setup. Packet tracing helps us to identify the problem layer, too. Example 4-9 shows a sample output of the Linux tcpdump tool. The tcpdump tool is used to capture the network packets on a system.

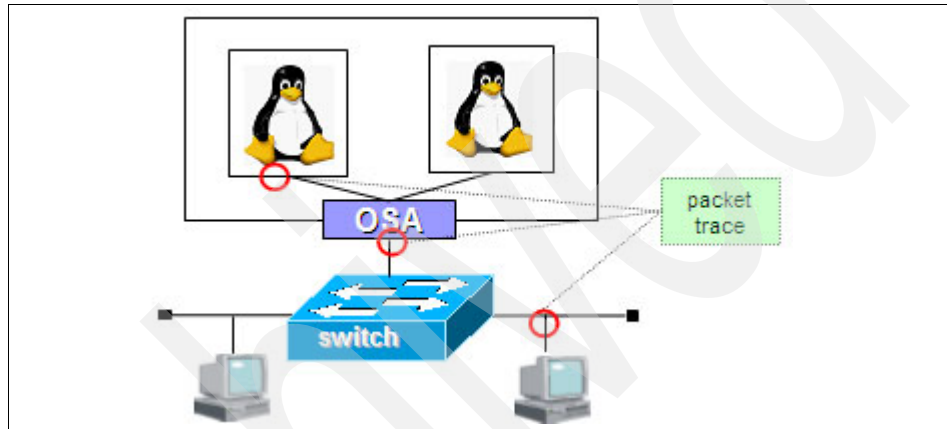


Figure 4-49 Using packet trace for problem determination

Example 4-9 Sample tcpdump output

```
# tcpdump -i eth0
20:41:10.810464 IP 172.168.1.2.ibm3494 > lin.itso.pok.ibm.com.ssh: .
ack 1144296 win 63403
20:41:10.810464 IP 172.168.1.2.ibm3494 > lin.itso.pok.ibm.com.ssh: .
ack 1144296 win 65535
20:41:10.810465 IP 172.168.1.2.ibm3494 > lin.itso.pok.ibm.com.ssh: .
ack 1145092 win 64739
20:41:10.810466 IP 172.168.1.2.ibm3494 > lin.itso.pok.ibm.com.ssh: .
ack 1145748 win 64083
20:41:10.810467 IP 172.168.1.2.ibm3494 > lin.itso.pok.ibm.com.ssh: .
ack 1145748 win 65535
20:41:10.810748 IP lin.itso.pok.ibm.com.sshIP lin.itso.pok.ibm.com.ssh
> 172.168.1.2.ibm3494: P 1149028:1149624(596) ack 5461 win 9648win 9648
20:41:10.811267 IP 172.168.1.2.ibm3494 > lin.itso.pok.ibm.com.ssh: P
5461:5513(52) ack 1145748 win 65535
```

```

20:41:10.811410 IP lin.itso.pok.ibm.com.ssh > 172.168.1.2.ibm3494: P
1149624:1150476(852) ack 5513 win 9648
20:41:10.811466 IP lin.itso.pok.ibm.com.ssh > 172.168.1.2.ibm3494: P
1150476:1150752(276) ack 5513 win 9648
20:41:10.811572 IP lin.itso.pok.ibm.com.ssh > 172.168.1.2.ibm3494: P
1150752:1151044(292) ack 5513 win 9648
20:41:10.811722 IP lin.itso.pok.ibm.com.ssh > 172.168.1.2.ibm3494: P
1151044:1151208(164) ack 5513 win 9648
20:41:10.811814 IP lin.itso.pok.ibm.com.ssh > 172.168.1.2.ibm3494: P
1151208:1151372(164) ack 5513 win 9648
20:41:10.811904 IP lin.itso.pok.ibm.com.ssh > 172.168.1.2.ibm3494: P
1151372:1151536(164) ack 5513 win 9648
20:41:10.811993 IP lin.itso.pok.ibm.com.ssh > 172.168.1.2.ibm3494: P
1151536:1151700(164) ack 5513 win 9648

```

During a packet trace exercise, if we are not able to obtain the MAC address of the destination node, then that means that either the local node is not able to send ARP requests or the destination is not able to respond to ARP requests. If a ping is not working, then either the local system is not able to send ICMP echo or the destination is not able to reply to the same.

CISCO SPAN port function

CISCO switches have a SPAN port function, as shown in Figure 4-50. The unicast packet sent to B from A can be copied and sent to the SPAN port. The span port provides the switch port analyzer function.

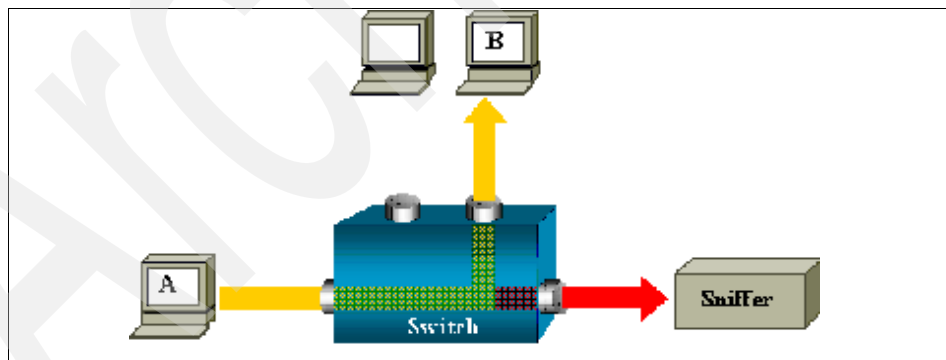


Figure 4-50 CISCO SPAN function

Packet tracing in z/VM

It is possible to capture the packets on the z/VM Guest LAN or VSWITCH by using the virtual LAN sniffer function (available in z/VM V5.2 or later). The CP TRSOURCE command can be used to trace and record the data transfer. To

analyze the output, the IPFORMAT utility can be used. To use the z/VM Guest LAN trace function, class C or higher privilege is required.

Figure 4-51 shows a z/VM packet trace scenario. Assume that we need to capture the trace information between the edge system and the IHS system.

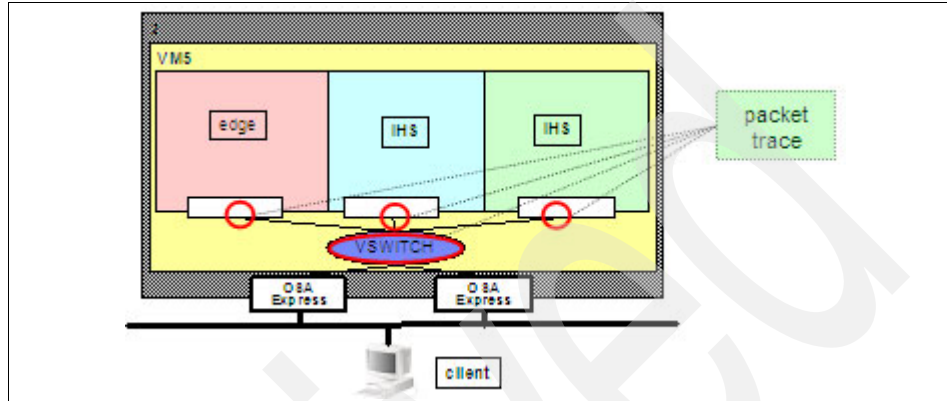


Figure 4-51 VM packet trace scenario

First we need to define the trace that we want to get. Then the defined trace needs to be activated. Once the trace is activated, we need to start the communication between the systems of interest. Once communication is attempted for a sufficient amount of time, the trace needs to be stopped. The next step is to save the trace information to a CMS file and then analyze the same using the IPFORMAT utility. Figure 4-52 shows the definition of the trace, activating the trace, and then starting the trace.

```

Ready, T=0.01/0.01 13:12:19
Step 1 → trso id vswitch1 type lan owner system lanname vswitch1
Ready, T=0.01/0.01 13:13:22
        q trso type lan

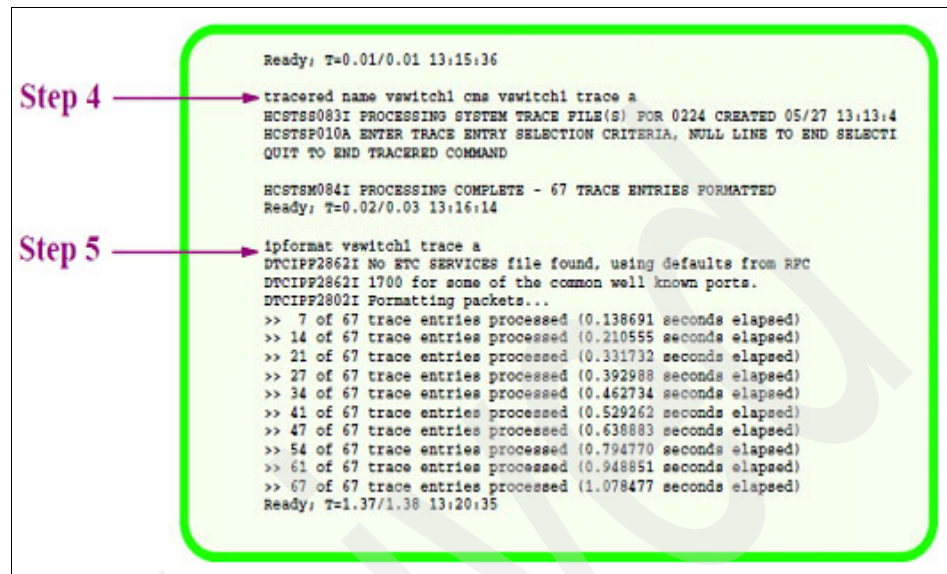
ID      TYPE      SET      STATUS  OWNER  NAME  LENGTH  OPTIONS
VSWITCH1 LAN      NULL    DISABLED SYSTEM VSWITCH1 0512  NONE
        VLANID
-      ALL
Ready, T=0.01/0.01 13:13:30
Step 2 → trso en id vswitch1
Ready, T=0.01/0.01 13:13:46
        q trso type lan

ID      TYPE      SET      STATUS  OWNER  NAME  LENGTH  OPTIONS
VSWITCH1 LAN      NULL    ENABLED  SYSTEM VSWITCH1 0512  NONE
        VLANID
-      ALL
Ready, T=0.01/0.01 13:13:50
Step 3 → trsource disa id vswitch1
Ready, T=0.01/0.01 13:15:31

```

Figure 4-52 z/VM Guest LAN sniffing step 1–3

Figure 4-53 illustrates stopping the trace and saving the trace record on to a CMS file. Also shown is the IPFORMAT command execution.



The screenshot shows a z/VM console session with two steps highlighted by red arrows and labels. Step 4 points to the 'traced name' command, and Step 5 points to the 'ipformat' command. The output shows the completion of the trace and the formatting of 67 trace entries.

```
Ready; T=0.01/0.01 13:15:36

Step 4 → traced name vswitch1 cms vswitch1 trace a
HCSTSS003I PROCESSING SYSTEM TRACE FILE(S) FOR 0224 CREATED 05/27 13:13:4
HCSTSP010A ENTER TRACE ENTRY SELECTION CRITERIA, NULL LINE TO END SELECTI
QUIT TO END TRACERED COMMAND

HCSTSM004I PROCESSING COMPLETE - 67 TRACE ENTRIES FORMATTED
Ready; T=0.02/0.03 13:16:14

Step 5 → ipformat vswitch1 trace a
DTCIPF2862I No ETC SERVICES file found, using defaults from RPC
DTCIPF2862I 1700 for some of the common well known ports.
DTCIPF2802I Formatting packets...
>> 7 of 67 trace entries processed (0.138691 seconds elapsed)
>> 14 of 67 trace entries processed (0.210555 seconds elapsed)
>> 21 of 67 trace entries processed (0.331732 seconds elapsed)
>> 27 of 67 trace entries processed (0.392988 seconds elapsed)
>> 34 of 67 trace entries processed (0.462734 seconds elapsed)
>> 41 of 67 trace entries processed (0.529262 seconds elapsed)
>> 47 of 67 trace entries processed (0.638883 seconds elapsed)
>> 54 of 67 trace entries processed (0.794770 seconds elapsed)
>> 61 of 67 trace entries processed (0.948851 seconds elapsed)
>> 67 of 67 trace entries processed (1.078477 seconds elapsed)
Ready; T=1.37/1.38 13:20:35
```

Figure 4-53 z/VM Guest LAN sniffing step 4–5

Figure 4-54 shows the trace record file contents. The contents are in the hexadecimal format. The IPFORMAT utility processes this record and generates a more easy-to-read output.

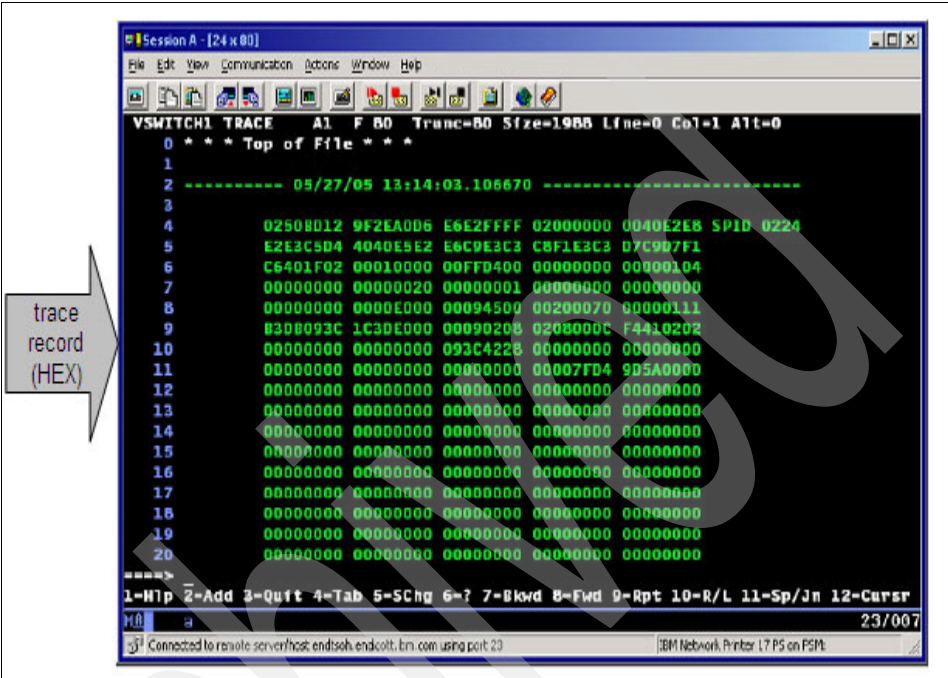


Figure 4-54 Sample trace record

Figure 4-55 shows the trace record after processing using the IPFORMAT utility.

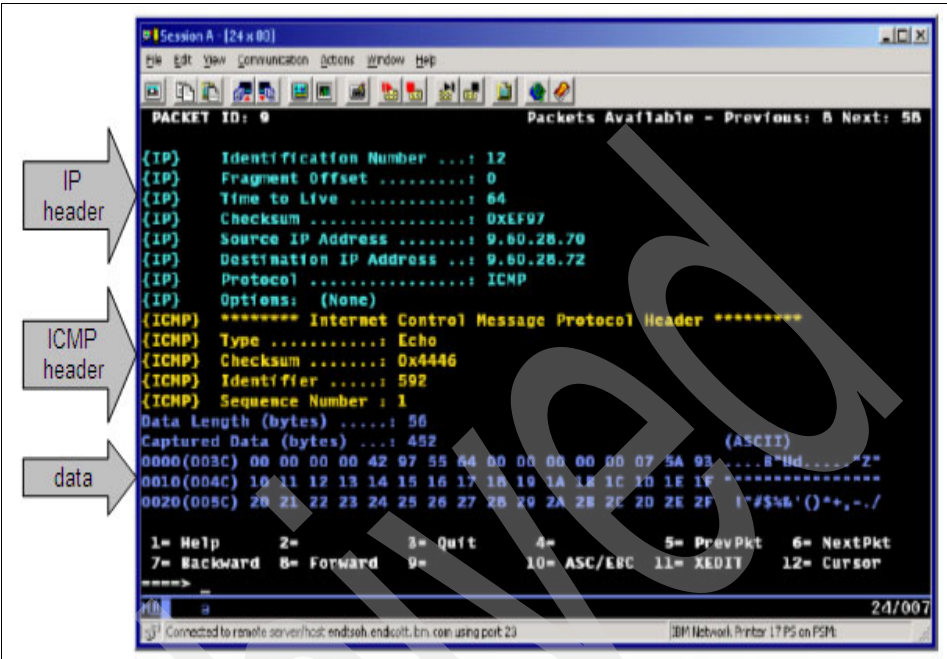


Figure 4-55 Sample IPFORMAT output

Example of a problem determination using packet trace

Figure 4-56 shows an example scenario for problem determination by using packet trace.

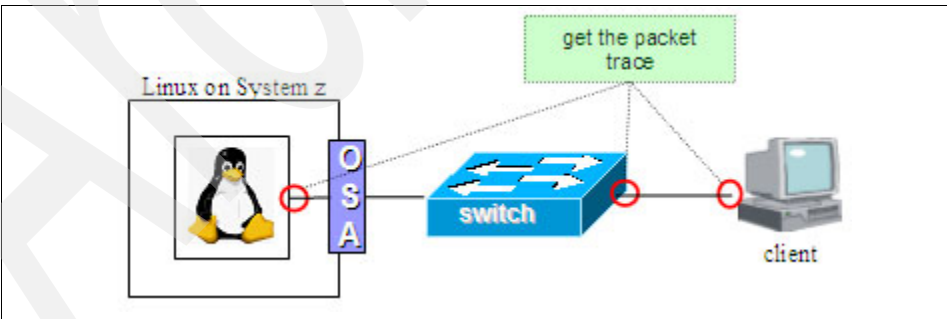


Figure 4-56 Example scenario for problem determination by using packet trace

Assume that the communication between the Linux system and the client is not proper. When a session is opened with the Linux machine from the client, the display is frozen for a while before returning to its normal state. Also, command

types are displayed on the screen after a while. To find the problem, as illustrated, we need to get Linux system trace information for the Linux system, switch and the client system.

As shown in Figure 4-57, we can see that the client and the switch are doing TCP retransmission continuously, but the Linux system is not seeing those packets. So, the client is not having any problem and the switch port is unstable. There is probably a speed mismatch between the OSA port and the network port setting. The OSA/SF tool can be used to find the OSA card port settings such as the negotiation mode, speed, and so on.

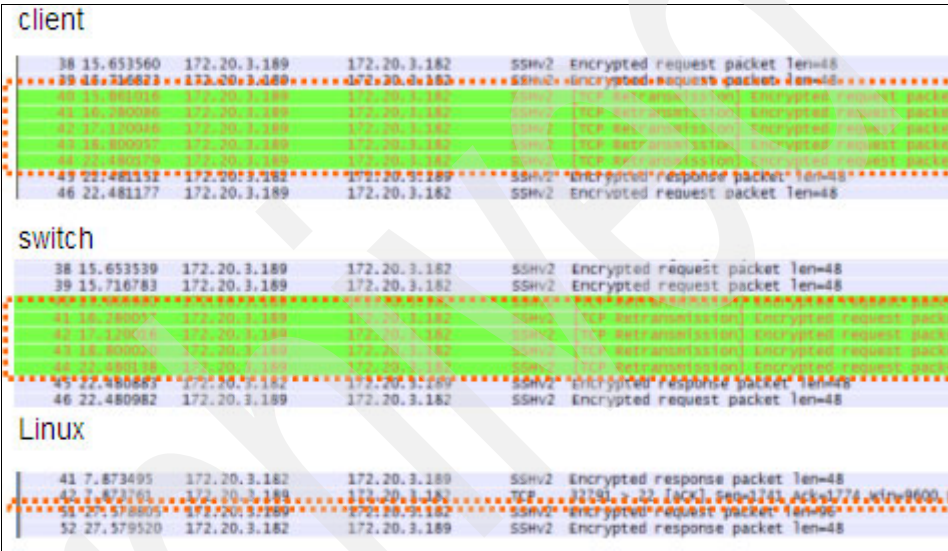


Figure 4-57 Example packet trace (switch port trouble)

This information can be compared with the switch-side settings to ensure that both are configured and operating in the same mode. As shown in Figure 4-58, the OSA/SF tools show the configured values for the OSA card and the active values.

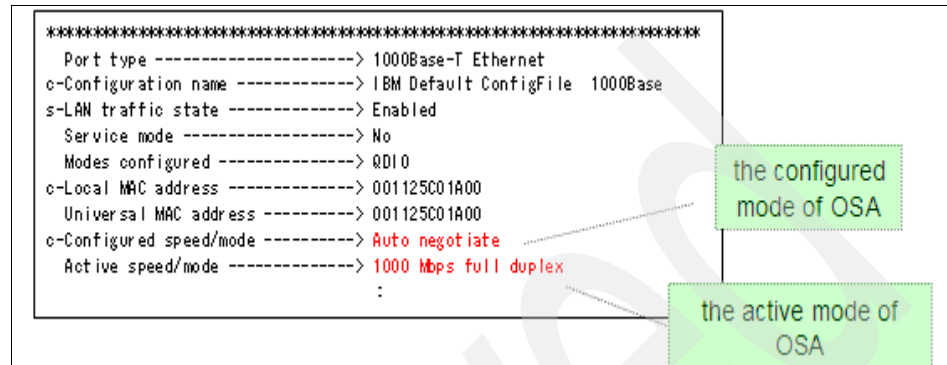


Figure 4-58 Sample OSA/SF output showing the OSA card port speed settings

4.4 Case studies

In this section we see some case studies involving real-life problems.

Case study 1 - eth0 interfaceeth0 interface down

In this scenario the deactivation of the eth0 interface causes the eth1 interface to stop functioning.

Problem

We have a Linux system running on the system with one interface eth0. A new Ethernet interface eth1 was added and the system was rebooted. After the system reboot, the eth0 interface was brought down using the **ifconfig** command to test the communication using the eth1 interface. But now the eth1 interface cannot communicate. Figure 4-59 shows the scenario.

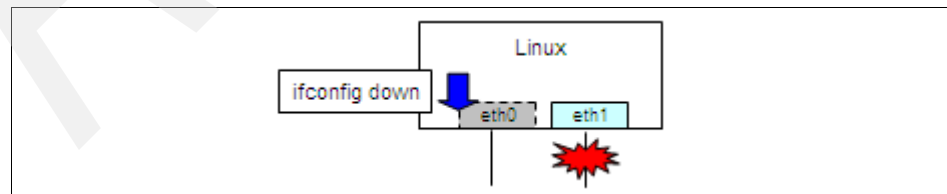


Figure 4-59 Case study example - eth0 interface down

Data for analysis

To troubleshoot the problem, we need to gather the following information and analyze it for correctness:

- ▶ `/etc/sysconfig/hardware/hwcfg-qeth-bus-ccw-0.0.xxxx`
- ▶ `/etc/sysconfig/network/ifcfg-qeth-bus-ccw-0.0.xxxx`
- ▶ `dmesg` command output
- ▶ `ifconfig` command output
- ▶ `/etc/sysconfig/network/routes`
- ▶ `netstat -rn` command output

Analysis and conclusion

Upon checking the configuration files, all looked correct. But from the routing table output given by the **`route -n`** command, we identify that the `eth0` interface is configured as the routing interface (default gateway) for `eth1`. So, when the `eth0` interface went down, the `eth1` interface was not able to communicate because of the loss of the routing table entries. All communication was done via the `eth0` interface.

If the above approach is not leading to a solution, then consider obtaining a packet trace.

Case study 2 - connection lost after VSWITCH OSA takeover

Assume that we have a VSWITCH configured on a z/VM system with device 3700 as the active OSA and device 3800 as the backup OSA.

Problem

When the cable connected to the active OSA (device 3700) was removed, the communication was lost even though switching to the backup OSA (device 3800) happened.

Data for analysis

Analyze the following data:

- ▶ The status of devices 3700 and 3800
- ▶ ARP cache of z/VM TCP/IP and the workstation executing the **`ping`** command
- ▶ Routing z/VM Linux guest information for the z/VM Linux guest (**`route -n`** command output)
- ▶ Trace information for the VSWITCH
- ▶ OSA/SF query

Analysis and conclusion

From the OSA/SF query output, it was found that the port setting of the backup OSA device was 1000 Mbps full duplex in spite of connecting to the 100 Mbps switch. Due to the mismatch of the port speed, data was not sent. Since the active OSA was able to communicate before the cable was pulled, it is clear that there is no problem at or above the IP layer.

Case study 3 - not able to connect using SSH

Consider the scenario shown in Figure 4-60. The Linux systems are running on z/VM on System z. The routers indicate that the client system and the Linux systems are in different network subnets.

Problem

We are not able to connect to the Linux system from the client using SSH.

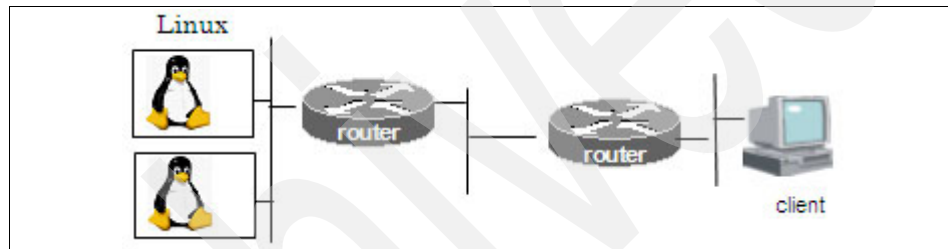


Figure 4-60 Case study example - not able to connect using SSH

Data for analysis

Analyze the following data:

- ▶ `/etc/sysconfig/hardware/hwcfg-qeth-bus-ccw-0.0.xxxx`
- ▶ `/etc/sysconfig/network/ifcfg-qeth-bus-ccw-0.0.xxxx`
- ▶ routing table information (`route -n` command output)
- ▶ configuration of routers
- ▶ `tcpdump`

Analysis and conclusion

There was no problem with PING or FTP. Since SSH will use the same network path as PING and FTP, the problem is not with the routing tables. SSH between the two Linux systems is working well. So, the SSH server on the Linux system is working correctly. Checking the router configuration, it was discovered that the MTU size setting of the router connected to the Linux system network segment is 512. This means that packet fragmentation is needed at the router level. But the client is sending packets with the Do not Fragment (DF) bit set. This causes the router connected to the Linux systems to send the message ICMP Destination unreachable/fragmentation needed and DF bit was set to the client. But the

configuration of the router connected to the client segment is to discard this packet. Hence, the cause of the problem is the configuration at the router side. The tcpdump was useful to troubleshoot this problem.

Performance problem determination

This chapter discusses performance problem determination for Linux on System z. We discuss:

- ▶ What a performance problem is
- ▶ General flow of performance problem determination
- ▶ Case studies

5.1 What a performance problem is

Typically, when the user of a system is expecting the system to deliver something and if the system is not able to deliver up to the expectation, then we say that there is a performance problem with that system. Based on the expectation of the user, this can vary. For example, when a system is running at 100% processor utilization and if the users do not face any issues with the responsiveness or throughput of the system, then there is no performance problem with the system. So, it is important to understand the expectation before trying to troubleshoot.

The majority of performance problems arise due to the misconfiguration of the system or not meeting all the prerequisites. Performance problems may arise because of resource bottlenecks (for example, applications or middleware consume all the resources on the system), software problems (improper configurations, software bugs, and so on), and hardware problems.

5.2 General flow of performance problem determination

Performance problems can arise at the hardware level, the operating system level, the middleware level, or the application level. Figure 5-1 shows the approach to problem determination classified in these layers. The scope of this chapter is problem determination at an operating system level. However, the following points should be noted:

- ▶ Investigation at the OS layer alone may not be always sufficient.
- ▶ Sometimes investigation at each separate layer is not enough, and the other layers needs to be investigated as well.

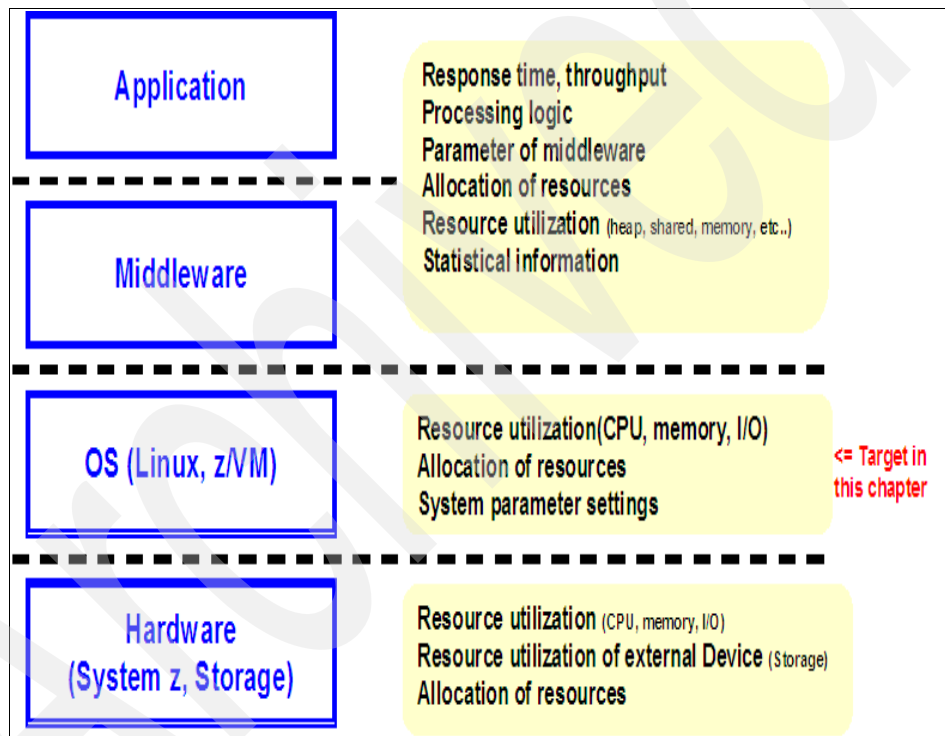


Figure 5-1 Performance problem determination layers

CPU, memory and I/O are all related to hardware, but these are the resources managed by the operating system, so the operating system level problem determination is typically concentrated around these resources. As illustrated in Figure 5-2, performance problem determination starts with understanding the problem and its occurrence.

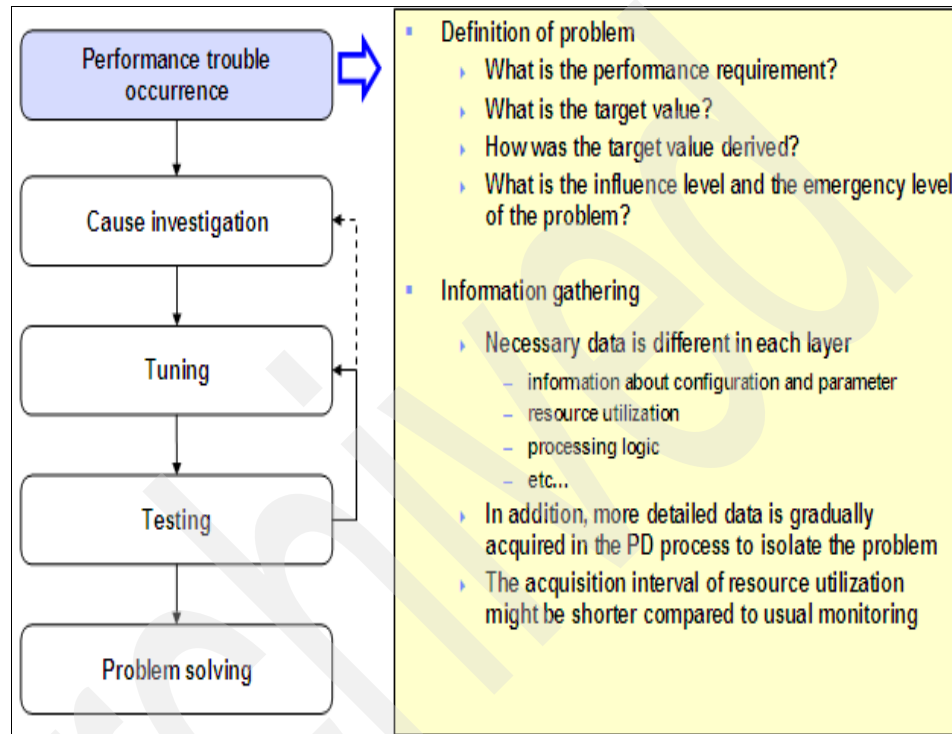


Figure 5-2 Performance problem determination - defining the problem

As shown in Figure 5-2, the first step is to understand the problem clearly. As noted, it is important to understand the expectations of the user of the system and what makes him think that there is a performance problem. It is also important to understand the system environment to learn about any external factors affecting the performance of the system. After understanding the problem description, the next step is to gather more information, such as configuration parameters, how resources are utilized, and processing logic of the applications. As the problem determination exercise progresses, more data is gathered for detailed analysis and to isolate the problem. It is also typical that during the problem-determination process, the system resource utilization is monitored very frequently for smaller intervals than what is done under normal circumstances.

As indicated in Figure 5-3, from an operating system perspective, the utilization of CPU, memory, disk, and network is the main data analyzed for problem determination. Although it depends on the problem at hand, in most of the cases the investigation starts by examining the CPU constraints. One reason for this is that the CPU constraint is probably the easiest to find.

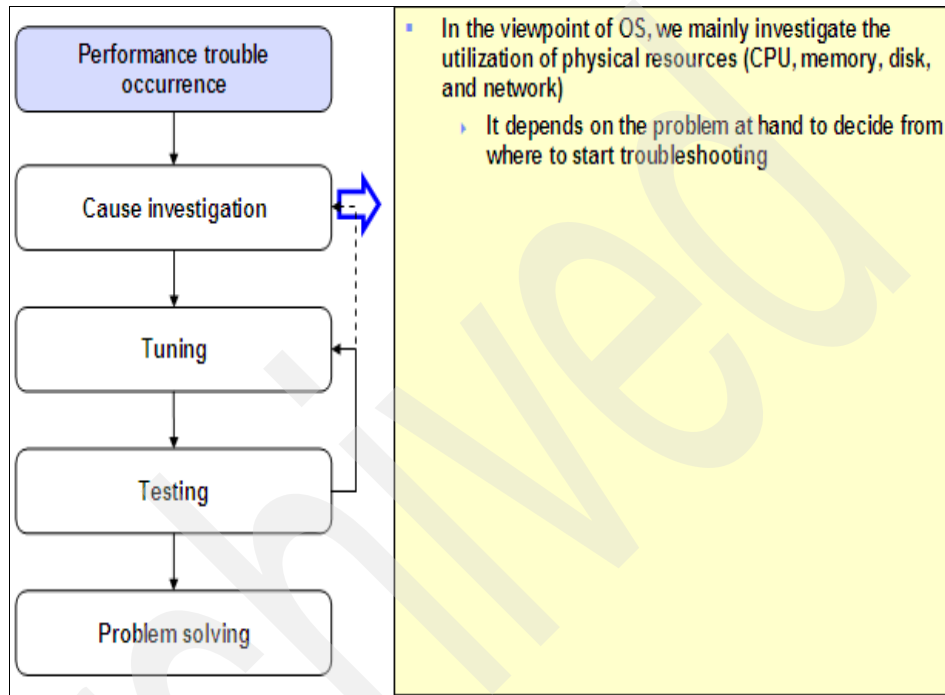


Figure 5-3 Performance problem determination - cause investigation

As shown in Figure 5-4 on page 168, the system is checked to see whether there are any CPU constraints. If the system seems to be CPU bound, then it is investigated further to understand which process and which application is creating the problem. If the system is not having CPU constraints, the next resource to be considered is memory. If there is a memory constraint, then it is investigated further to understand exactly where this is happening. If CPU and memory constraints are not found, the next thing to examine is an I/O resource constraint. In case of I/O resources, there are two major sub-resources: disk and network. If there is no resource constraint observed at the I/O level either, then it may be necessary to investigate the interaction of different layers indicated in Figure 5-1 on page 165. Sometimes it may happen that solving one bottleneck will introduce another bottleneck, which needs to be resolved.

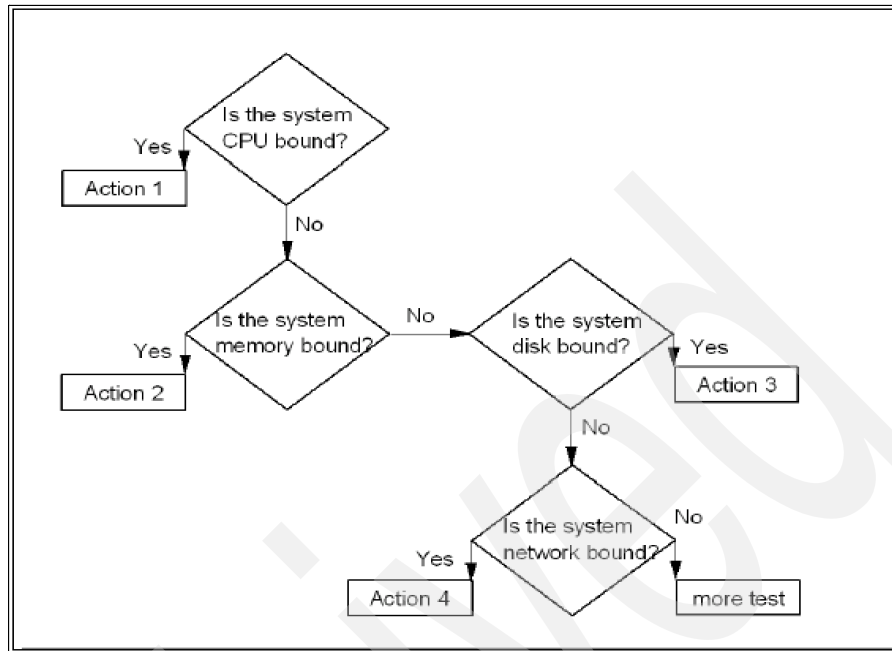


Figure 5-4 Performance problem determination - typical cause investigation flow

Generally, problems occur because of a change. This could be a change in the system configuration, a change in the resource allocation, a change in the parameters used for the configuration, and so on. When problems occur in an environment where everything was working perfectly, the first thing to be checked is for any change (in the environment, on the system, and so on).

As shown in Figure 5-5, investigation, tuning, and testing are interrelated activities. This is because an investigation will lead to a solution, and it is important to test the solution to make sure that it really works. Also, when the solution is tested, it may become apparent that the solution is solving the problem at hand but introduces a new bottleneck. In this case, the newly identified problem has to be investigated and a new solution needs to be found.

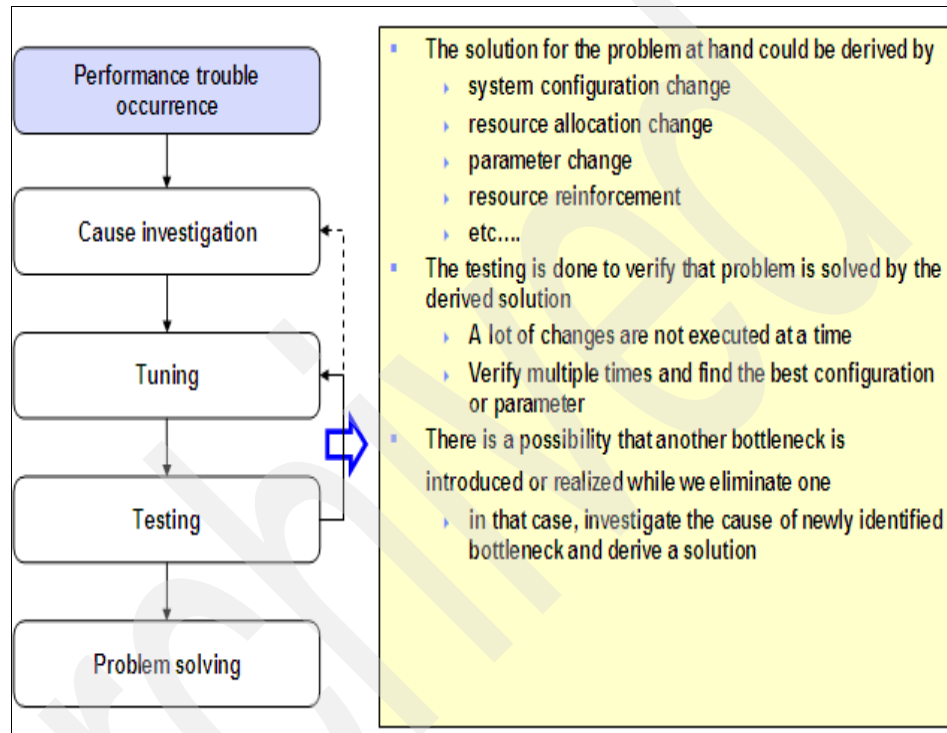


Figure 5-5 Performance problem determination - tuning and testing

While testing a solution, it is important to make only one change at a time. When many changes are done at the same time, it will be difficult to conclude which change solved the problem, if it gets resolved. Making multiple changes at the same time might introduce new problems as well. Then it becomes difficult to roll back since, we may not know which change created the new problem. It is a good practice to verify a solution multiple times to find the best configuration or parameter.

5.2.1 Data necessary for investigation

Collecting as much data as possible is important for any kind of problem determination. This is true for performance problem determination, too. We need

to collect the information about the system configuration such as the mode of operation (that is, LPAR or z/VM guest), how many CPUs are allocated, how much memory is allocated, how many disks are assigned, minidisks or dedicated disks, how many channel paths are there for disk access, type of networking options used (QDIO, iQDIO, and so on), software versions (kernel level, z/VM service level, and so on), and type of applications running on the system and their nature (that is, CPU intensive, memory intensive, I/O intensive, and so on). We need to understand how the system is configured, such as the kernel parameters configured and resource usage limits (for example, ulimit). Another important set of information is to collect the resource utilization data for the system under consideration. Resource utilization data includes CPU utilization, memory utilization, disk utilization, and network utilization. In the case of CPU and memory, what we need to know is how much CPU or memory there is, how much is utilized, how quickly its is getting consumed, who is using it, and so on. In the case of disk and network, what we need to know is how much data is getting transferred, how many I/O operations are performed to transfer the data, and so on. Although it is useful to have real-time statistics, for effective performance problem determination, historical data for the resource utilization is required. In case historical data is not available, resource utilization should be measured at regular intervals and the results should be analyzed.

Refer to Chapter 3, “Problem determination tools for Linux on System z” on page 59, for a summary of the common Linux tools used for information gathering.

5.2.2 CPU bottlenecks

In this section we discuss CPU bottlenecks. Let us first take a brief look at how Linux and z/VM uses the CPU.

The Linux CPU management is illustrated in Figure 5-6. Only one process gets executed on a given CPU at a time. The processes that are ready to be executed on a CPU are maintained in a ready queue. From there, the Linux system puts a process to the processor. After some time, the Linux system moves out that process from the CPU and puts in another process. The processes waiting to enter the ready queue are maintained in a waiting queue.

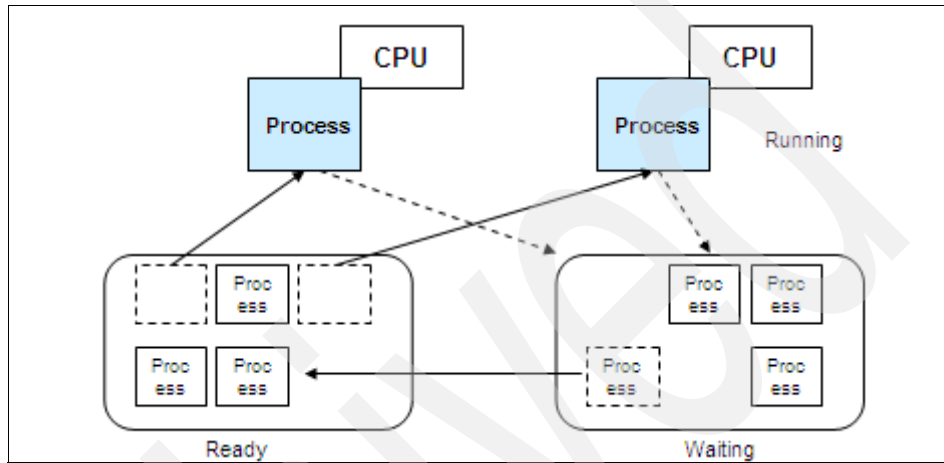


Figure 5-6 Linux CPU management

Figure 5-7 illustrates the CPU sharing on System z. At the LPAR level, the physical CPUs are assigned to logical CPUs by the logic in the microcode. The processes executed by each LPAR on the logical CPUs assigned to them are executed on the physical CPUs on a time-sharing basis by the microcode logic. In case of z/VM, the virtual machines see virtual CPUs assigned by z/VM. Similar to the microcode logic, z/VM takes care of executing the processes from the virtual machines on a time-sharing basis.

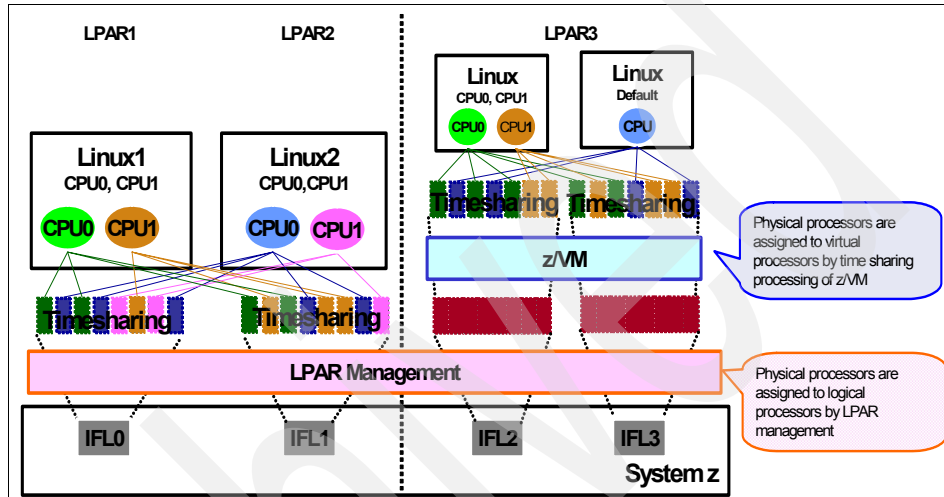


Figure 5-7 CPU sharing on System z

The CPU utilization is calculated as shown in Figure 5-8. As seen in figure, the CPU time is not the entire duration. It is only those times during which the CPU was really used. For example, when the `vmstat` command is executed with a 5-second interval, the tool calculates how many times the CPU was used during this 5-second interval.

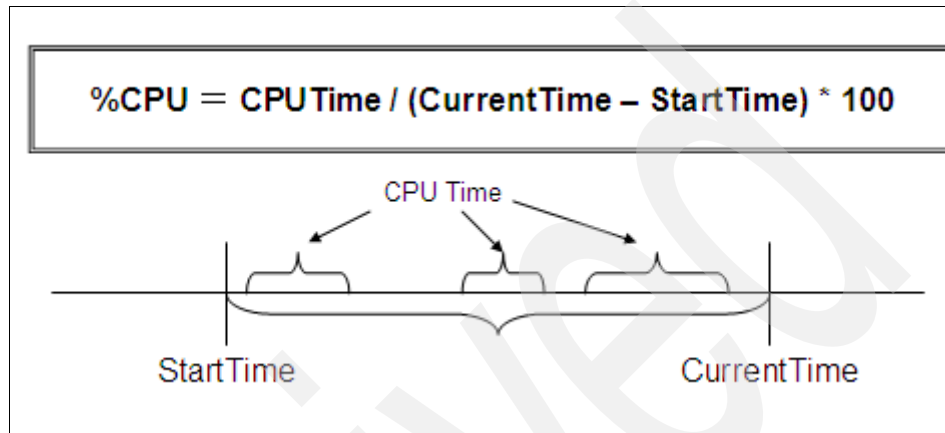


Figure 5-8 CPU utilization

As discussed earlier, we start the problem determination by examining the system configuration. Figure 5-9 shows the logic flow for the CPU bottleneck check. Typically, while examining the current configuration, the physical, logical, and virtual (z/VM) CPU configuration is learned. Then the CPU resource allocation settings are compared with the physical CPU resources on the hardware to make sure that the configuration is correct.

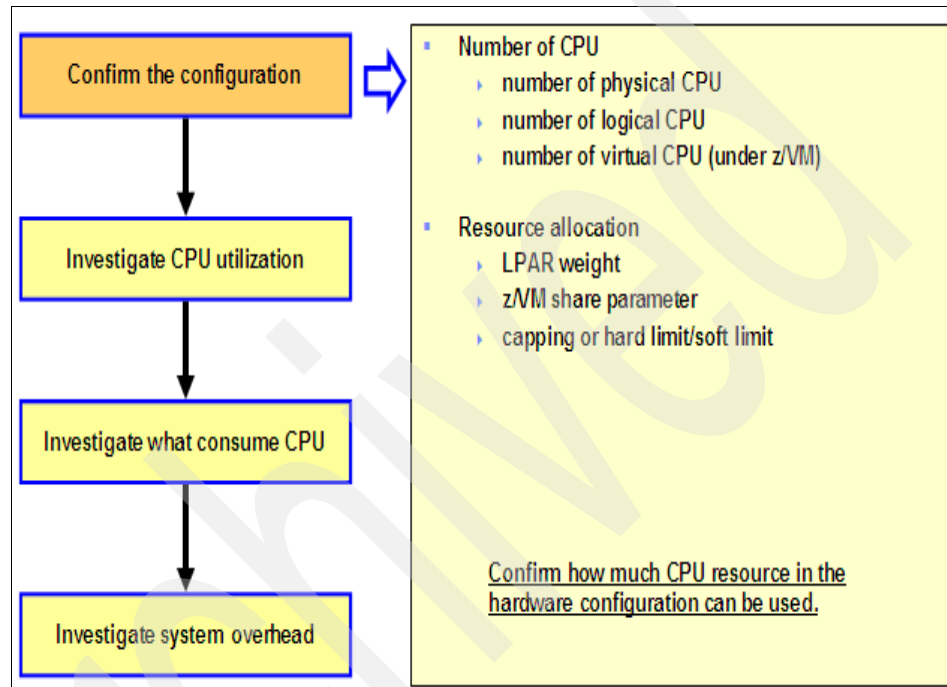


Figure 5-9 Flow of problem determination - checking the configuration

Once the configuration is verified, the next step is to investigate the cause of the high CPU utilization problem. Figure 5-10 on page 175 shows a diagrammatic representation for the same. When the CPU utilization is investigated, it is important to remember that the utilization data needs to be collect at multiple intervals and then analyzed. The CPU utilization data must be collected for the entire system, for each CPU, and for each process. In the case of z/VM, the same is to be collected for each VM and, also, the z/VM overhead is to be measured. If the CPU utilization is 100%, then chances are that the bottleneck is on the CPU. However, this is not always true since activities such as paging consume lot of processing cycles. So, if the CPU utilization is reaching 100% mainly because of the paging operations, then we have a memory bottleneck.

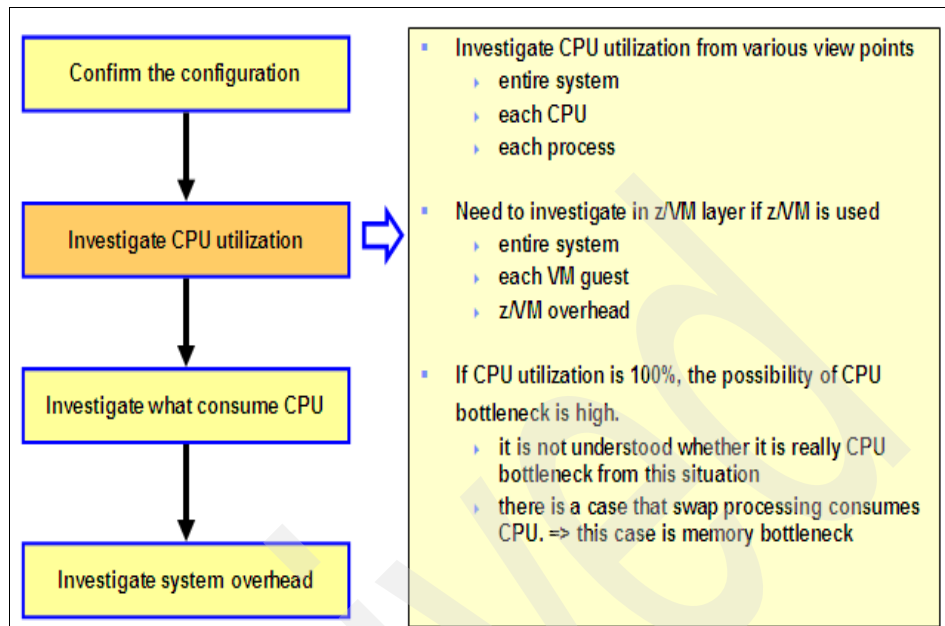


Figure 5-10 Flow of problem determination - investigating the CPU utilization

Table 5-1 summarizes the tools that can be used for acquiring the CPU utilization under Linux. This is not a complete list—these are some of the common tools.

Table 5-1 Tools to acquire CPU utilization under Linux

Name	Summary	Entire system	Each CPU	Each process
vmstat	Acquire CPU utilization at specified interval.	X		
mpstat	Acquire CPU utilization at specified interval.	X	X	
sar	Acquire CPU utilization at specified interval.	X	X	
top	Acquire CPU utilization of each process.	X	X	X

Name	Summary	Entire system	Each CPU	Each process
ps	Acquire CPU utilization of each process.			X

z/VM Performance Toolkit is capable of acquiring the CPU utilization for the entire system, for each CPU, and for each process.

Table 5-2 Tools to acquire CPU utilization under z/VM

Name	Summary	Entire system	Each CPU	Each process
Performance Toolkit	Acquire CPU utilization of z/VM and each VM guest.	X	X	X

Figure 5-11 shows a sample vmstat output where the CPU utilization of the entire system is available.

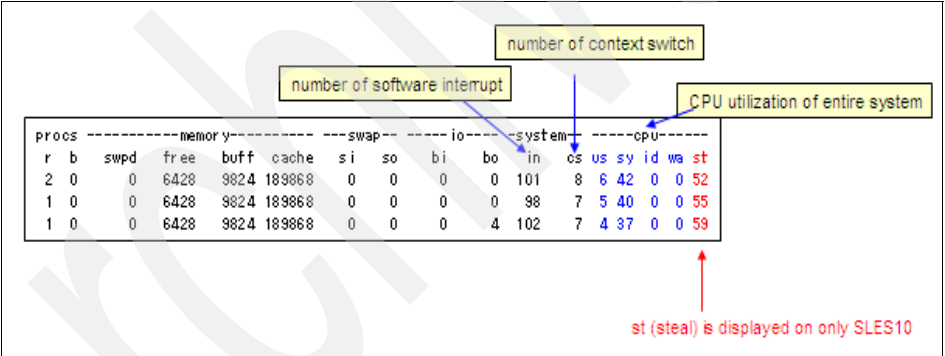


Figure 5-11 CPU utilization of entire system - vmstat

Figure 5-12 shows a sample sar output where the entire CPU utilization, as well as the utilization of each CPU is available.

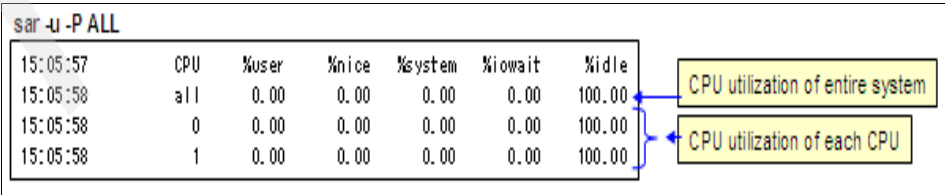


Figure 5-12 CPU utilization of entire system - sar

Figure 5-13 shows a case where from the vmstat output it looks like there is no CPU bottleneck, but the output of sar and mpstat indicate that there is a CPU bottleneck. In multiprocessor systems such as the one in this case, the vmstat output will be an average of each CPU utilization. So, the vmstat output shown in Figure 5-13 indicates that the system is having only 50% CPU utilization. However, the sar and mpstat outputs indicate that one processor is always at 100%, and hence there is a bottleneck.

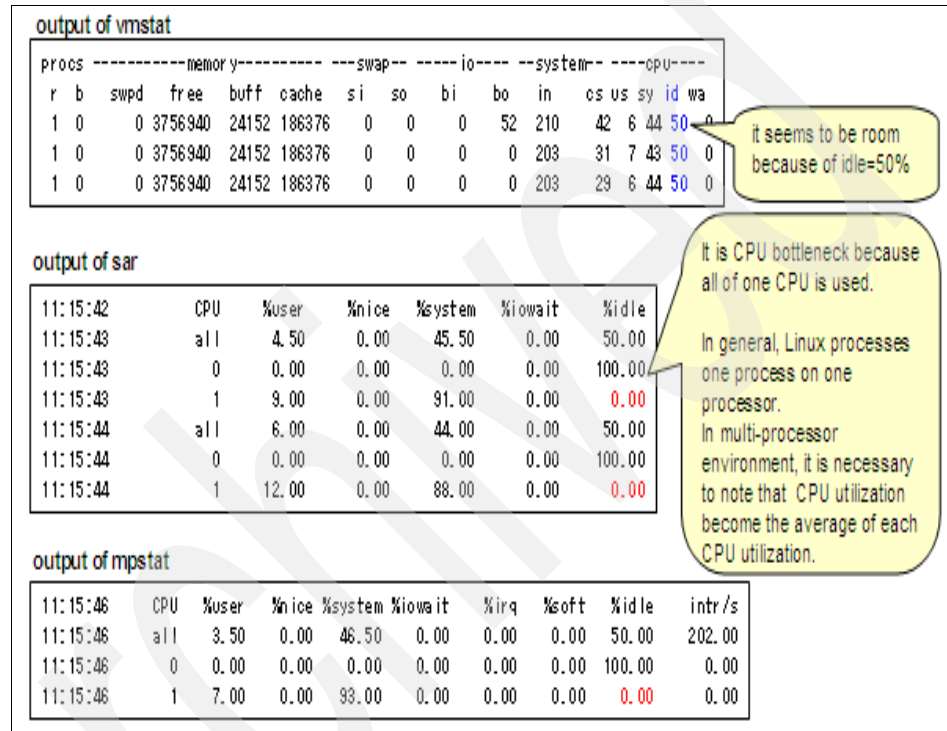


Figure 5-13 Utilization of each CPU

The **top** and **ps** commands can be used to find the individual processes using the CPU. These commands are capable of listing the CPU utilization for each process. Figure 5-14 on page 178 shows sample outputs for **top** and **ps**. As can be seen from the figure, the **top** command is capable of displaying the CPU utilization of each process on which CPU the process is running, how much CPU time is used by each process, and the utilization of each CPU.

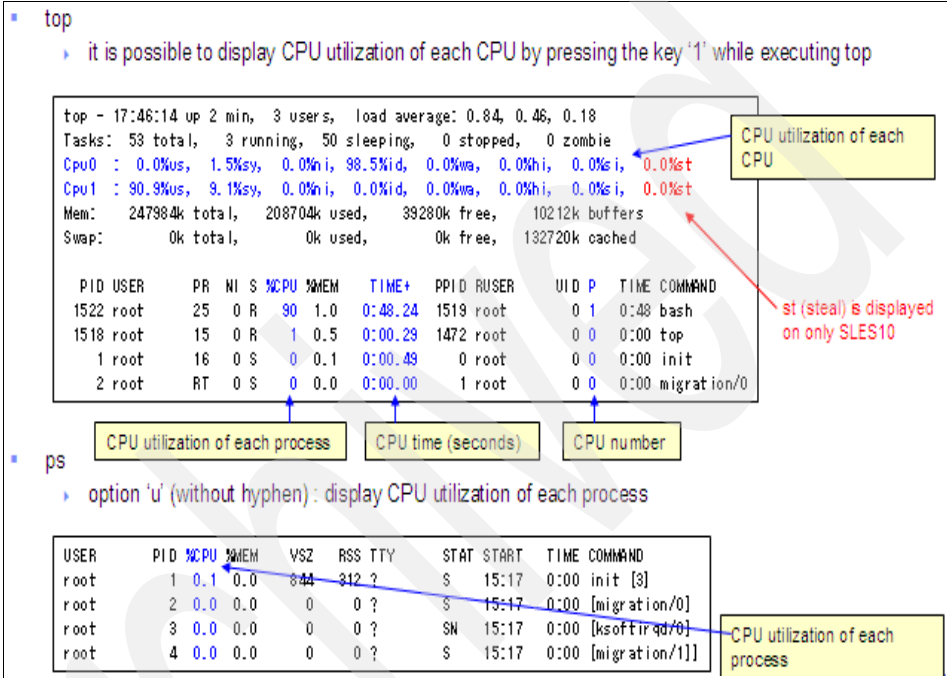


Figure 5-14 CPU utilization of each process

FCX000 Run 2005/04/19 11:15:16 CPU
General CPU Load and User Transactions

From 2005/04/07 14:38:16
To 2005/04/07 14:37:16
For 60 Secs 00:01:00 "VMFF sample"

CPU Load										Vector Facility			Status or
PROC	%CPU	%CP	%EMU	%WT	%SYS	%SP	%SIC	%LOGLD	%VTOT	%VEMU	REST	ded. User	
P00	85	27	58	15	0	0	100	85	0	0	0	
P01	29	6	22	71	0	0	99	29	0	0	0	

Total SSOH/RSOH	5/s	Page rate	31/s	Priv. instruct.	47598/s
Virtual I/O rate	5411/s	XSTORE paging	6.3/s	Diagnose instr.	47549/s
Total rel. SHARE	4300	Tot. abs SHARE	0%		

Queue Statistics:	Q0	Q1	Q2	Q3	User Status:
VMDBs in queue	9.0	0.0	0.0	1.0	# of loaded on users
VMDBs loading	0.0	0.0	0.0	0.0	# of dialled users
Eligible VMDBs	0.0	0.0	0.0	0.0	# of active users
El. VMDBs loading	0.0	0.0	0.0	0.0	# of in-queue users
Tot. WS (pages)	829061	0	0	0	% in-Q users in PQMALT
Expansion factor	% in-Q users in IQMALT
85% elapsed time	8.552	1.069	8.552	51.31	% elig. (resource wait)

Transactions	Q-Disp	trivial	non-triv	User Extremes:
Average users	0.0	0.0	0.0	Max. CPU % CMS1
Trans. per sec.	0.3	0.0	0.0	Max. VECT %
Av. time (sec)	.121	.000	.000	Max. IO/sec CMS1
IP trans. time		.000	.000	Max. PGS/s LNK02
MF trans. time		.000	.000	Max. RESP LNK04
System ITR (trans. per sec. tot. CPU)			.4	Max. MCLIO
Emul. ITR (trans. per sec. emul. CPU)			0.0	Max. XSTORE LNK01

Depending on the Performance Toolkit report screen, the CPU utilization is displayed as an average of all CPUs or a total of all CPUs as follows:

- Chapter 5. Performance problem determination
- 179**

To further investigate the problem, it is necessary to find which process is consuming the CPU. In case of Linux, this could be a user process or a kernel thread. Figure 5-16 summarizes the action points in each case.

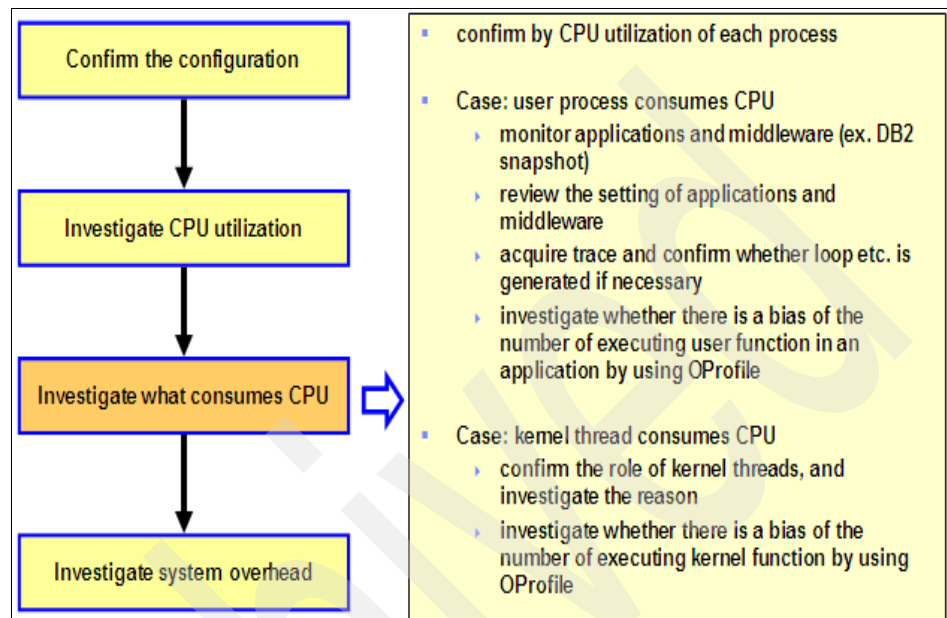


Figure 5-16 Flow of problem determination - investigate what consumes CPU

kswapd is a Linux kernel thread that acquires pages when the free memory decreases. So, a high CPU utilization by kswapd means that there is memory shortage and CPU is getting consumed because of that.

kjournald is a Linux kernel thread that commits the transaction of file systems using a journal such as the ext3 file system. High CPU utilization by kjournald indicates that a lot of CPU is being consumed for file system transaction commit to the journal area due to lot of disk I/O.

pdflush is a kernel thread that writes dirty page cache to the disk. High CPU utilization by pdflush indicates that there is lot of disk I/O since it is actually writing dirty page caches to the disk very often.

When software IRQ load on the Linux system increases, the ksoftirqd kernel thread is executed. This is done on a per-CPU basis. So, if the ksoftirqd is taking more than a tiny percentage of CPU time, it means that the system is under heavy software interrupt load. This can happen because of very high disk I/O, heavy network traffic, and so on.

What happens if we do not find a specific process using a lot of CPU but the individual CPU utilization is reported as high? There is still a chance that the CPU is really getting utilized. The time taken by the CPU to perform the context switch is considered as the system time. This is the system overhead since it is not used by any process directly. An increase in this time means that more and more CPU is used for context switch. Other tasks that are considered as overhead are swap processing and a large amount of I/O. As summarized in Figure 5-17, the number of context switches, and swapping, large I/O are the reasons behind the increase in system overhead and hence high CPU utilization.

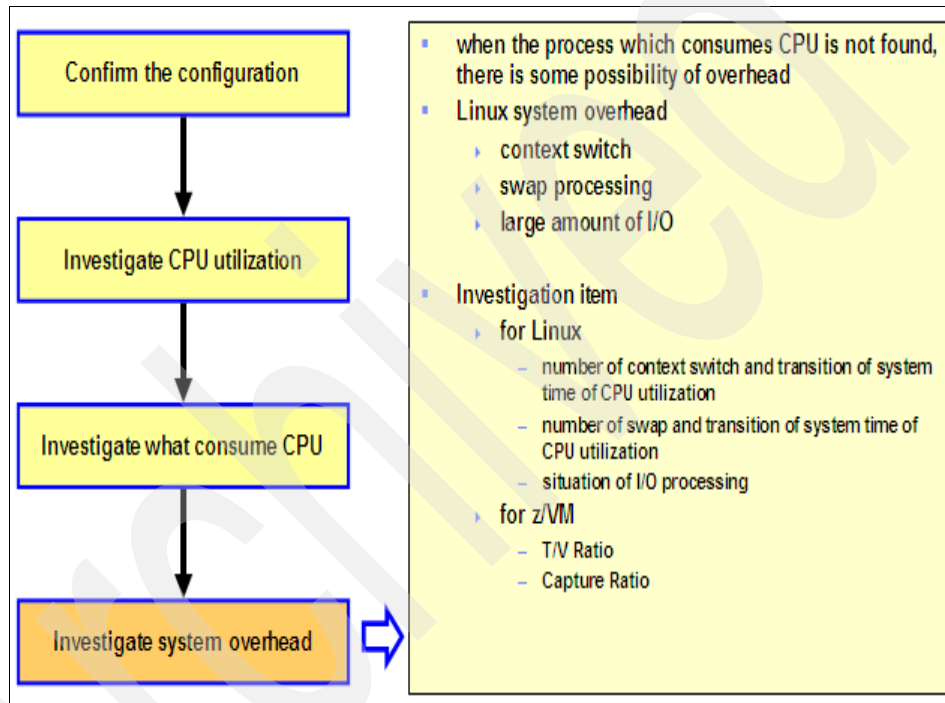


Figure 5-17 Problem determination flow - investigate system overhead

In case of z/VM the ratio of user time to emulation time or the service time (called as the T/V ratio) is a measurement that shows whether the user can efficiently use the resource. Similarly, the capture ratio is a measurement that shows whether the system can efficiently use the resource. The capture ratio is the ratio of user time to (system time + user time). This is illustrated in Figure 5-18.

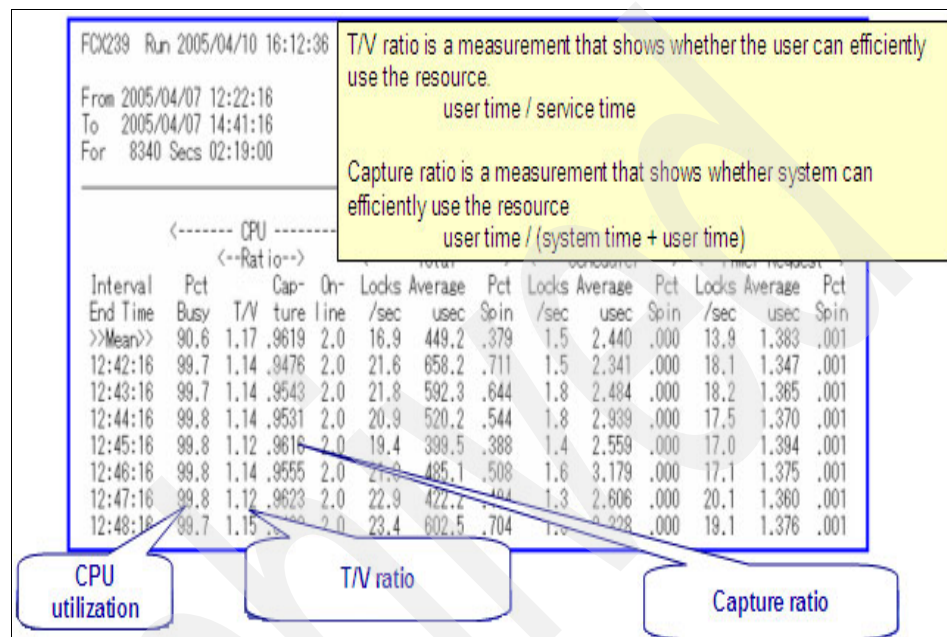


Figure 5-18 VM overhead check

5.2.3 Memory bottlenecks

Linux executes each process in a virtual address space. The translation from the virtual address to the real address is executed as a page based on the page translation table. The page size is fixed at 4 KB. This is illustrated in Figure 5-19. The virtual memory is a logical layer between the application memory requests and the physical memory.

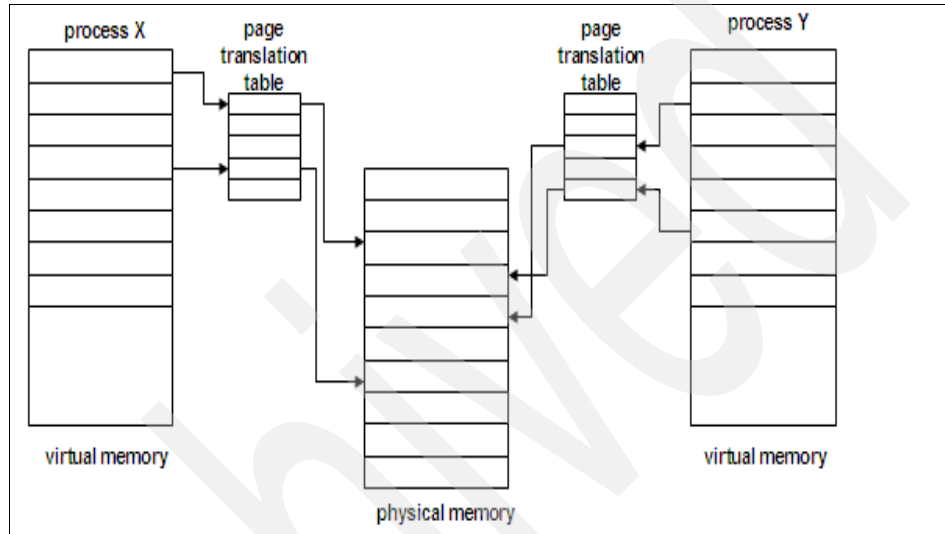


Figure 5-19 Linux memory management

When the main memory is not enough for the system to execute processes, the system will try to move some data from the main memory to the disk. This process is called swapping or paging. Swapping occurs when the system needs more memory that is available. But swapping has the downside of slowing down the system since the disk is a slow device compared to the main memory. So, accessing the disk for swapping will slow down the system.

When the swapped-out pages are to be accessed, they are swapped-in to the memory again. Figure 5-20 illustrates swapping.

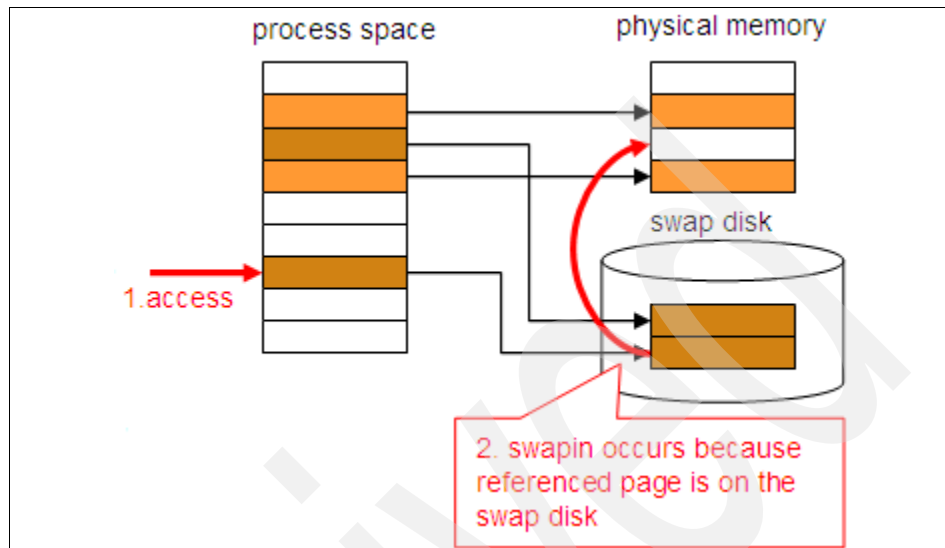


Figure 5-20 Swapping

In case of z/VM, there is a main storage area where the processes are executed. There is an expanded storage that is used for paging. DASD space is also used for paging. This is illustrated in Figure 5-21. The CS in the figure indicates main storage and the ES indicates expanded storage.

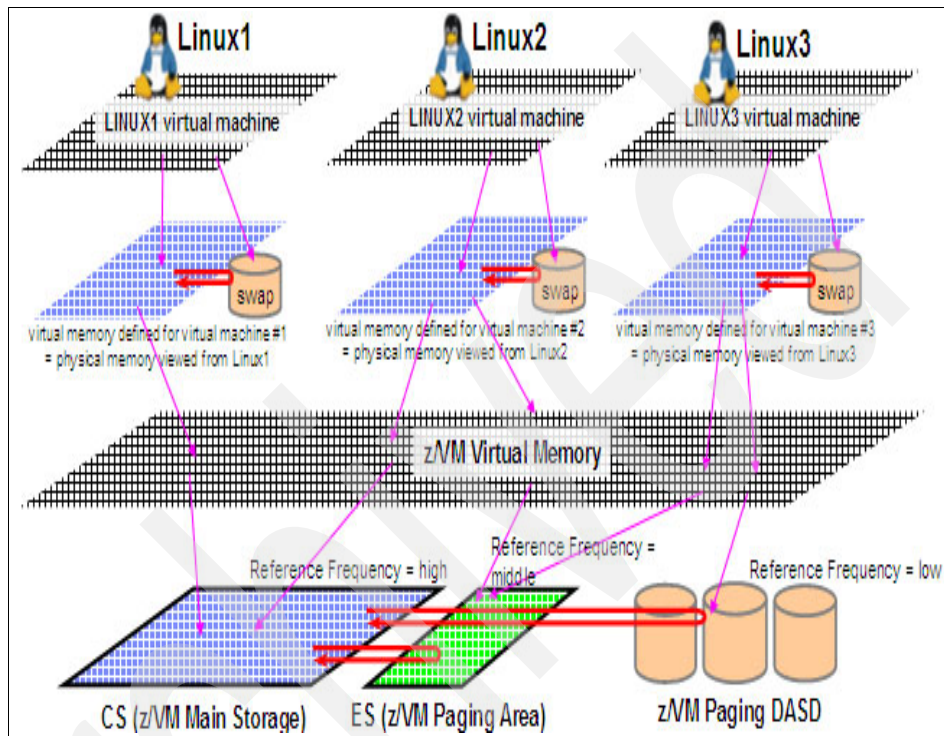


Figure 5-21 z/VM paging mechanism

Figure 5-22 on page 186 shows the memory bottleneck problem determination flow. As usual, first we start by examining the current configuration. Depending on the configuration, how much memory assigned as main storage, how much as expanded storage, virtual memory setting for each guest (z/VM), shared memory settings for middleware, and so on, are examined. When the same memory space is shared with multiple processes, then it is called as the shared memory area. Under Linux, this is typically used with middleware such as DB2®, Oracle, and so on. It is possible to control whether the shared memory should be swapped. The area in the memory where the data necessary for application execution is stored is called the JVM™ heap in the case of Java. Memory utilization of JVM cannot be viewed from the operating system and should be viewed from the JVM side.

The next step is to check whether swapping occurs. Figure 5-23 summarizes this.

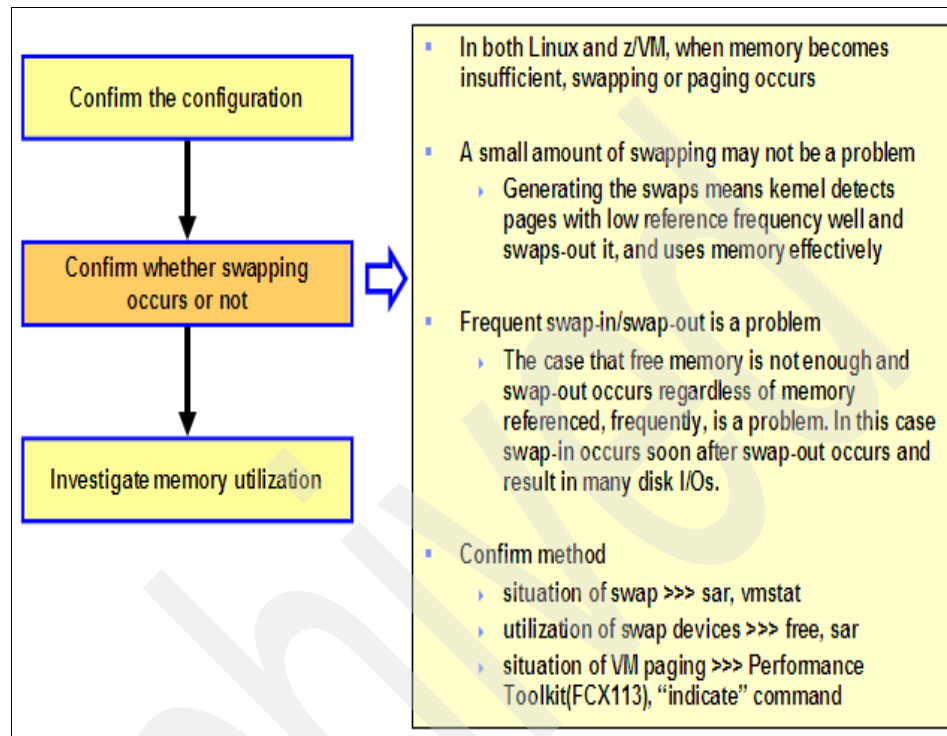


Figure 5-23 Problem determination flow - check for swapping

Figure 5-24 illustrates how to check whether swapping occurs from the Linux side by using the **free** command.

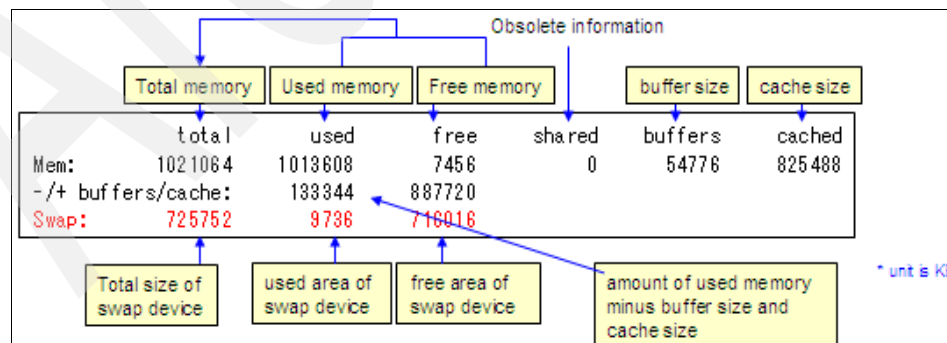


Figure 5-24 Confirming swapping by using free

Figure 5-25 illustrates how to check the paging activity from z/VM using the Performance Toolkit. Paging happens if there is a guest that gives overhead in the >2GB> memory area, when pages are migrated between the expanded and the main memory (X>MS and MS>X), and when pages are migrated between the expanded memory area and the disk storage area (X>DS). Large values for the Working Set Size (WSS) also cause paging.

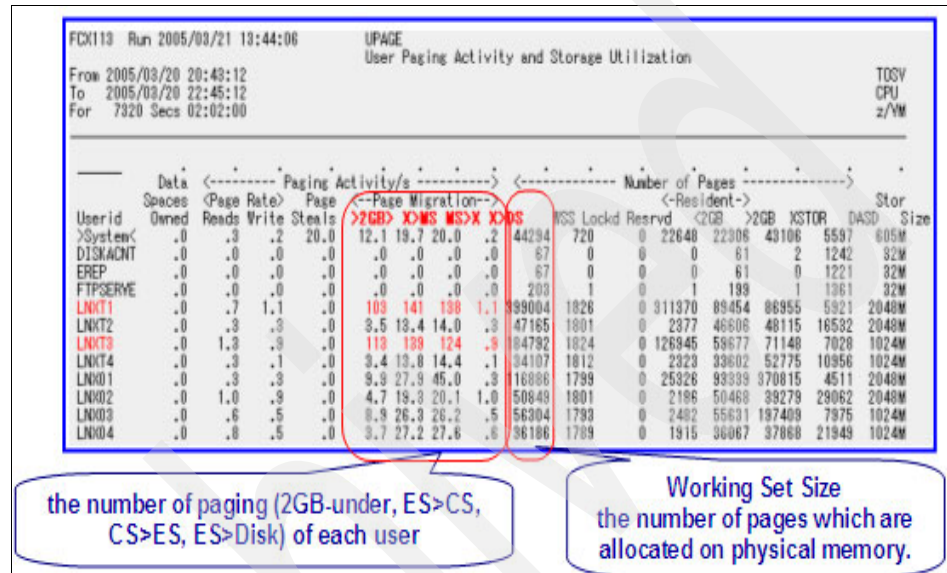


Figure 5-25 Confirming swapping by using z/VM Performance Toolkit

The next step is investigating where the memory is getting consumed. As shown in Figure 5-26, the memory may be getting consumed by a user process, shared memory areas, JVM heaps, file system cache, and so on. As discussed earlier in this chapter, it is not sufficient if we have only one snapshot of the memory utilization. We need such snapshots generated at regular intervals to understand the nature of the problem.

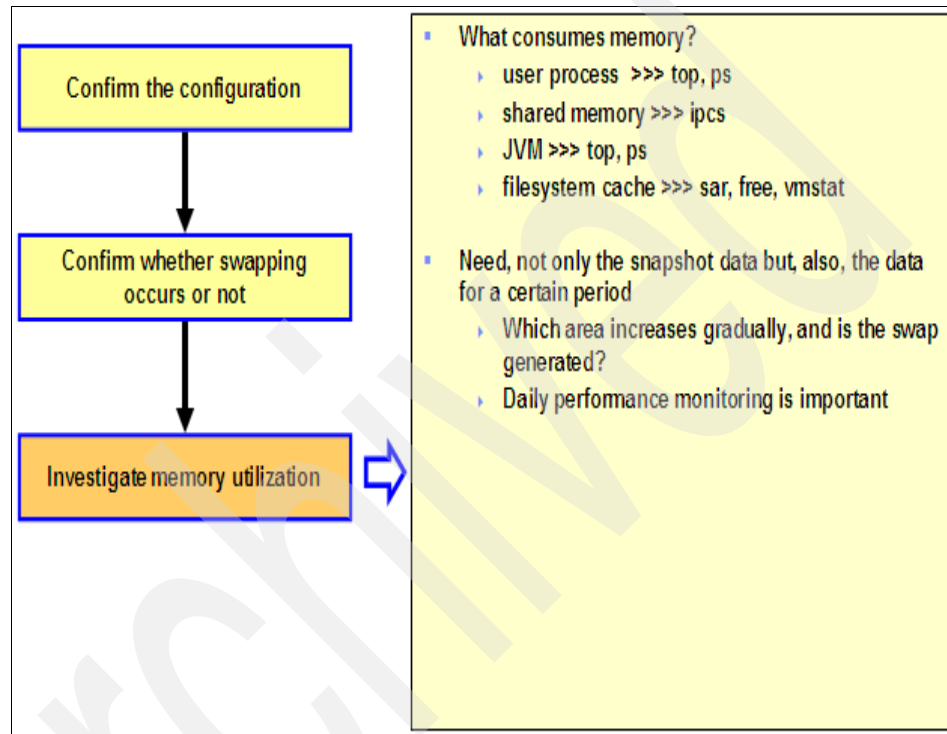


Figure 5-26 Problem determination flow - investigating the memory utilization

Figure 5-27 shows a sample top output with the output data related to memory explained.

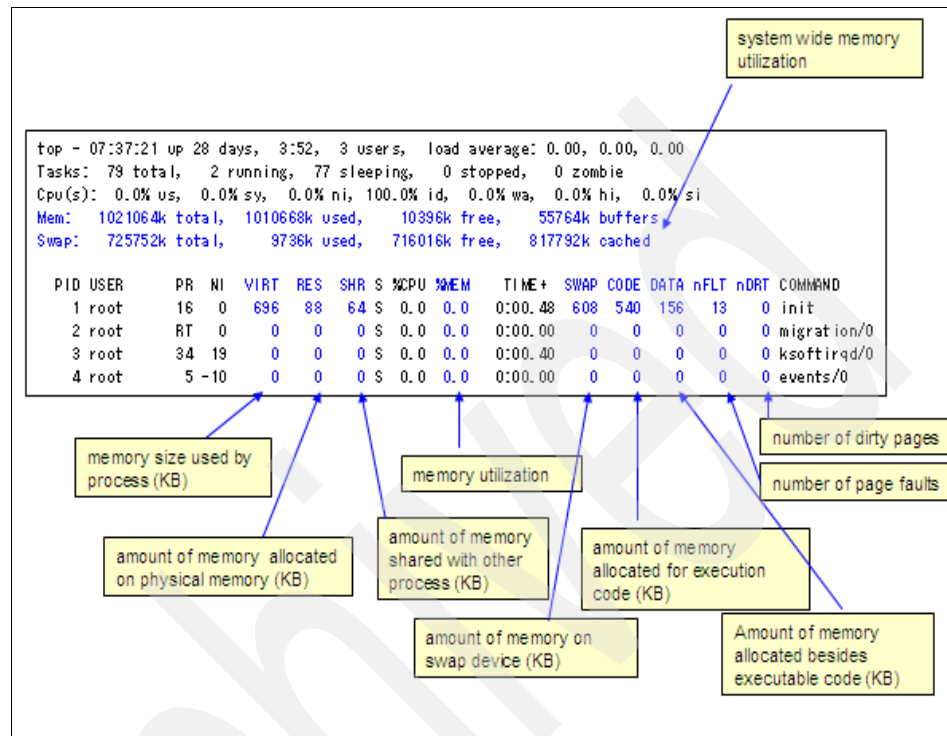


Figure 5-27 Checking memory utilization by using top

Figure 5-28 shows a sample ipcs output illustrating how to check the shared memory usage.

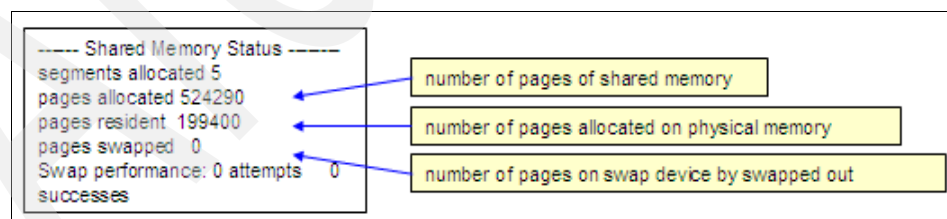


Figure 5-28 Checking memory utilization by using ipcs

Linux uses the free memory as the file system cache. So, it is normal to find that Linux does not have any free memory if it is running for a long period of time. Such a cache gets released when a processes is executed.

Figure 5-29 shows a sample output z/VM Performance Toolkit indicating general storage utilization. The output indicates the utilization of the usable area for paging in the main storage, the number of pages under the 2 GB memory area, and the utilization of the expanded storage, in addition to many other details. If the number of pages under the 2 GB memory area is above 1000, then it should be investigated.

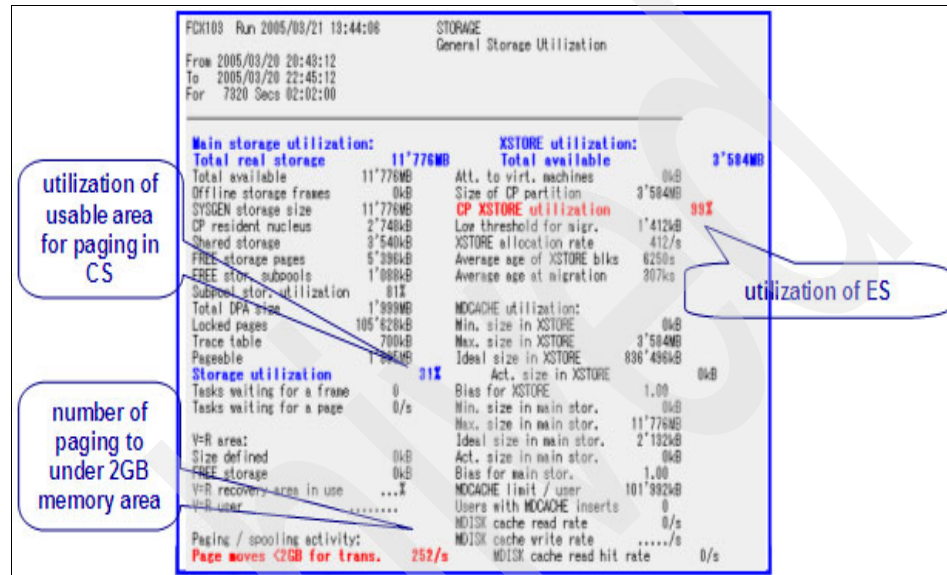


Figure 5-29 Confirming memory utilization by using z/VM Performance Toolkit

part of DB2 snapshot		part of Oracle Statspack				
Buffer pool data logical reads	= 370	SGA Target Size (M)	SGA Size Factor	Est. DB Time (s)	Est. DB Time Factor	Est. Physical Reads
Buffer pool data physical reads	= 54	5,632	.3	237,747	1.0	5,245,722
Buffer pool temporary data logical reads	= 0	11,284	.5	237,510	1.0	5,194,345
Buffer pool temporary data physical reads	= 0	18,898	.8	237,488	1.0	5,189,874
Buffer pool data writes	= 3	22,528	1.0	237,488	1.0	5,189,874
Buffer pool index logical reads	= 221	28,180	1.3	237,488	1.0	5,189,874
Buffer pool index physical reads	= 94	33,792	1.5	237,488	1.0	5,189,874
Buffer pool temporary index logical reads	= 0	38,424	1.8	237,387	1.0	5,183,207
Buffer pool temporary index physical reads	= 0	45,056	2.0	237,387	1.0	5,183,207
Total buffer pool read time (ms)	= 287					
Total buffer pool write time (ms)	= 1					
Asynchronous pool data page reads	= 9	SGA Memory Summary DB/Inst: xxxxxxxx Snap: 52-53				
Asynchronous pool data page writes	= 0					
Buffer pool index writes	= 0					
Asynchronous pool index page reads	= 0					
Asynchronous pool index page writes	= 0					
Total elapsed asynchronous read time	= 0					
Total elapsed asynchronous write time	= 0					
Asynchronous data read requests	= 3					

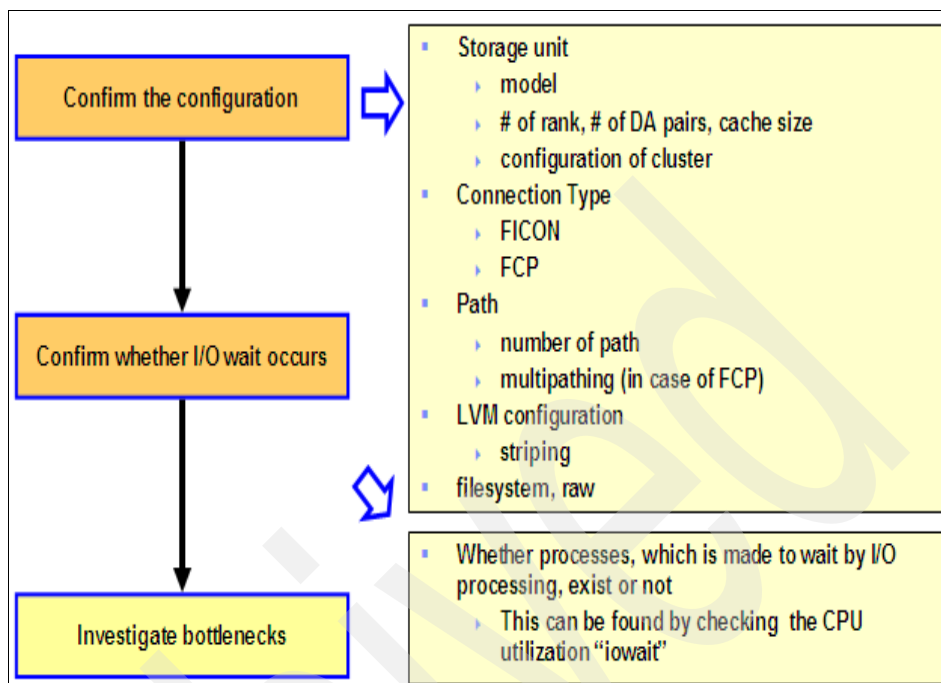


Figure 5-31 Disk bottleneck problem determination flow

There are two types of disks used with Linux on System z. One is the traditional mainframe DASD (ECKD™) provisioned using FICON®. The other one is the SCSI disk provisioned using FCP. So, the type of connection (that is, FICON or FCP) needs to be determined. The number of paths for the connection from the System z server to the storage subsystem is also important. In the case of FCP connection, multipathing information must be gathered. At the Linux side, the Logical Volume Manager (LVM) configuration must be examined, if in use. Also, information about the type of file systems or partitions used needs to be gathered.

Figure 5-32 shows how to use the `vmstat` command to check whether the CPU is made to wait for I/O operations to complete. It also shows how many processes are waiting for the I/O operations to finish.

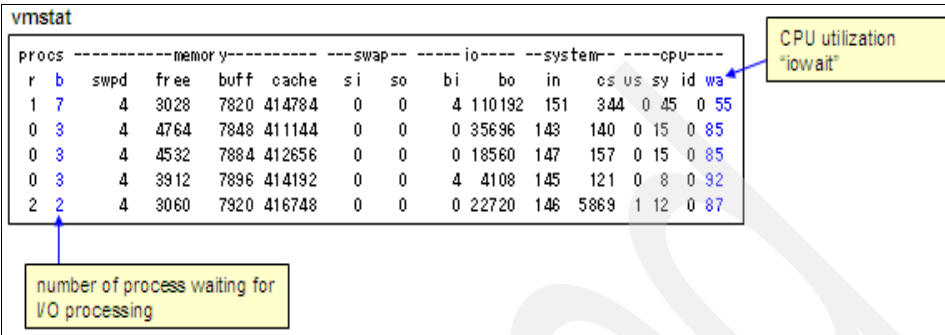


Figure 5-32 Examining disk bottleneck by using `vmstat`

Figure 5-33 indicates the points to be considered for investigation at the Linux side, the System z server side, and the storage subsystem side.

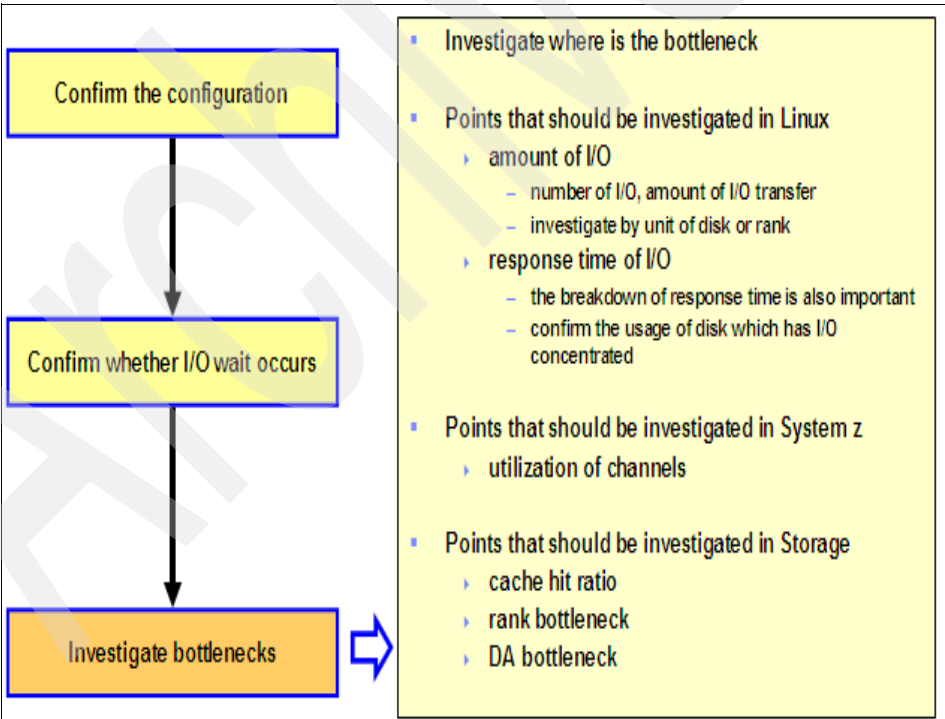


Figure 5-33 Problem determination - investigate the disk bottlenecks

Figure 5-34 shows a sample output of the **iostat** command. The output of **iostat** is for each device, which when collated will give the I/O detail for each rank of the storage subsystem. The number of I/O operations (IOPS) corresponds to r/s and w/s. The amount of data transfer corresponds to kB/s and kB/s. The average response time corresponds to await and svctm. If there is a big difference between await and svctm, there is lot of time spent in the device driver queue and channel is busy. If the svctm value is high, there is a bottleneck at the storage subsystem (the storage server) side. If the value of avgqu-sz is high, it means that there are many I/O operations happening and the waiting time is high as well.

Device:	rrqm/s	wrqm/s	r/s	w/s	rsec/s	wsec/s	rkB/s	wkB/s	avgrq-sz	avgqu-sz	await	svctm	%util
dasdek	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dasdej	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dasddv	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dasddv	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dasddz	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dasde	0.10	510.40	0.90	106.00	31.20	4931.20	15.60	2465.60	46.42	1.37	12.83	1.17	12.50
dasdf	0.00	0.00	0.00	0.10	0.00	0.80	0.00	0.40	8.00	0.00	10.00	10.00	0.10
dasdde	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dasdop	0.00	326.20	0.40	43.50	3.20	2957.60	1.60	1478.80	67.44	0.03	0.57	0.57	2.50
dasdi i	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dasdu	0.00	0.20	0.00	0.40	0.00	4.80	0.00	2.40	12.00	0.00	0.00	0.00	0.00
dasdhv	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dasdbv	0.00	2.10	0.00	0.30	0.00	13.20	0.00	3.60	64.00	0.00	3.33	3.33	0.10
dasddw	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dasdbf	0.40	179.00	7.20	25.80	1729.60	1638.40	864.80	819.20	102.06	0.06	1.85	1.61	5.30
dasdfc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dasdfd	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure 5-34 Investigating the disk bottlenecks by using iostat

Figure 5-35 shows the sample output of z/VM Performance Toolkit indicating the I/O device load and performance. The key parameters to look for are the Rate/s, Time (msec), and %Busy. The Rates/s value should not exceed the practical limit of the storage subsystem. The connect and service times in the Time (msec) column are expected to be nearly identical under normal circumstances. If the value of %Busy exceeds 50, there is a performance issue and the response time is high. For good performance, this value should be below 30.

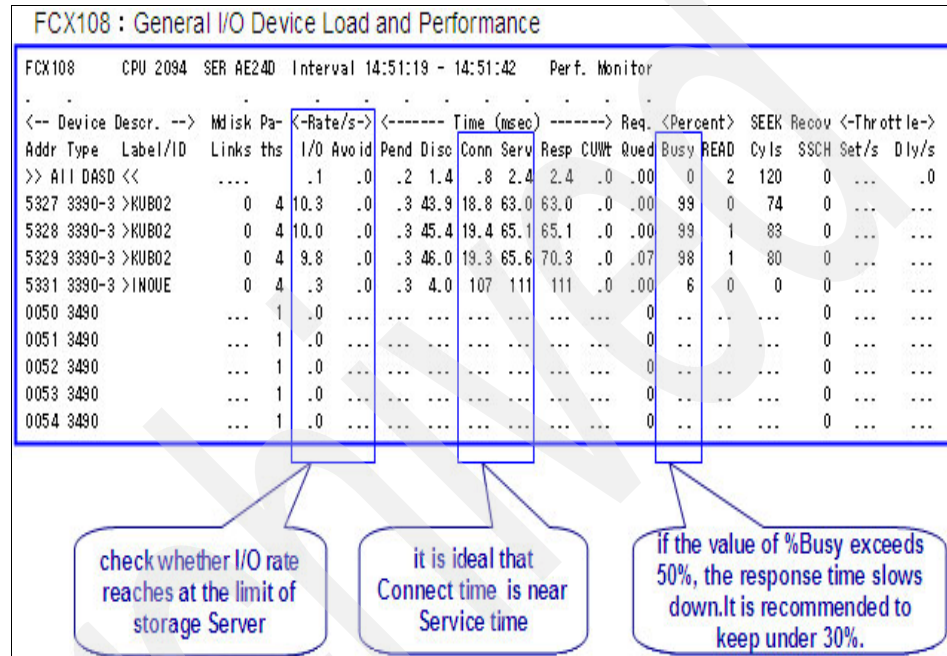


Figure 5-35 I/O load and performance check by using the z/VM Performance Toolkit

DASD statistics is a tool to monitor the activities of the DASD driver and the storage subsystem. It can be activated with the following command:

```
echo set on > /proc/dasd/statistics
```

It can be deactivated with:

```
echo set off > /proc/dasd/statistics
```


Refer to 3.1.9, “DASD statistics” on page 69 for a detailed explanation.
Figure 5-38 shows an example analysis of the DASD statistics data.

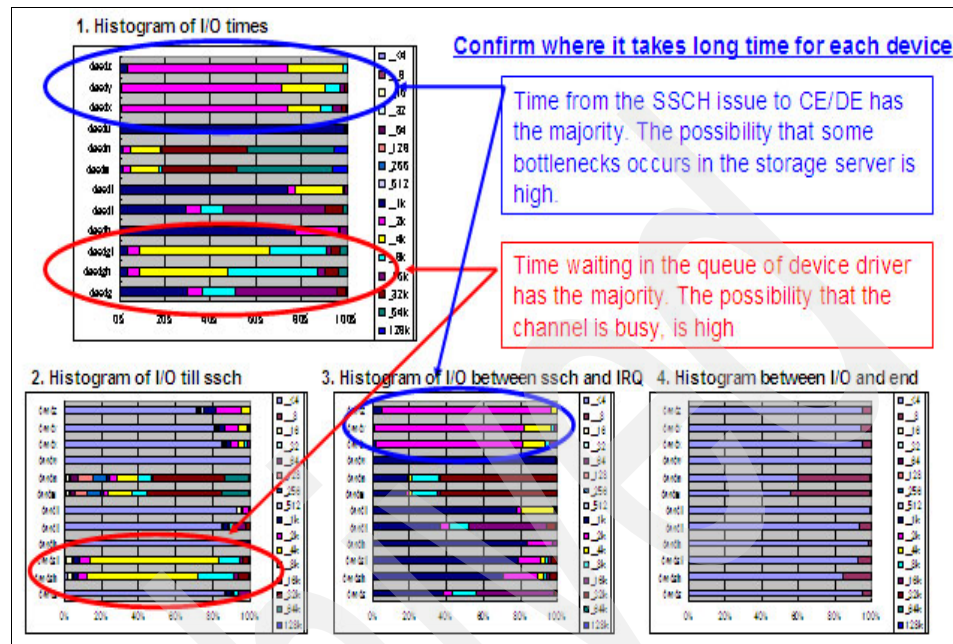


Figure 5-38 DASD statistics analysis example

The facility used to record I/O activities for FCP LUNs (SCSI disks) is zfcps statistics. The statistics of the Host Bus Adapter (HBA) and LUN can be acquired using this.

Figure 5-39 shows the breakdown of the SCSI latency that can be acquired using the zfcps statistics. Refer to 3.3.6, “zFCP/SCSI” on page 106 for a more detailed explanation. Table 5-3 indicates the conclusions to derive for an HBA based on the zfcps statistics.

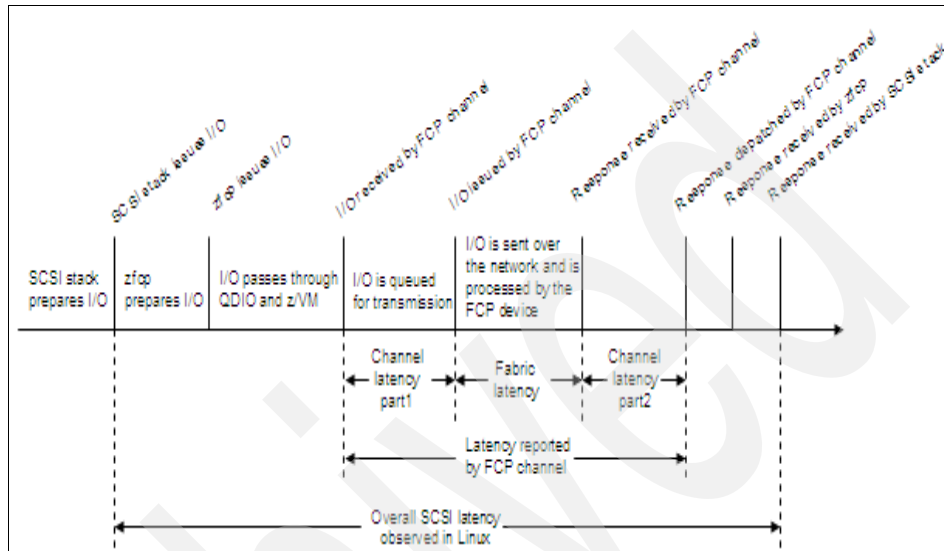


Figure 5-39 Breakdown of SCSI latency

Table 5-3 lists what look for on the host bus adapter based on the zfcps statistics.

Table 5-3 Statistics for host bus adapters

Description	What to look for
Number of queue full conditions for the request queue	Increase might indicate channel saturation.
Request queue utilization. Values: 0 to 128	Numbers close to 128 might indicate channel saturation.
Response queue utilization. Values: 0 to 128	Numbers close to 128 might indicate channel saturation.
Number of low memory conditions that require use of emergency buffer for SCSI commands	Increase indicates memory pressure, possibly caused by extreme I/O workload.
Number of error recovery actions of the HBA	Increase might indicate hardware problems.

Table 5-4 lists the some of the things to look for in the logical unit based on the zfcps statistics.

Table 5-4 Statistics for logical units

Description	What to look for
List of request sizes and their occurrences for SCSI write commands	Usually larger requests utilize the link better, but may increase device latency.
List of request sizes and their occurrences for SCSI read commands	
Number of SCSI commands without data transfer	Usually only used for recovery or discovery purposes, hence the number should be low.
Number of timed-out SCSI commands for write, read, and nodata commands	Increase might indicate hardware problems.
SCSI command sizes rejected due to subchannel queue constraints	Increase might indicate channel saturation.
SCSI commands rejected due to memory constraints	Increase might hint at workload or memory constraint as cause of performance degradations.
Ranges of latencies and their occurrences for SCSI write commands	High latencies indicate problems with the storage device, or the cabling.
Ranges of latencies and their occurrences for SCSI read commands	
Ranges of latencies and their occurrences for SCSI commands without data transfer	
Average, minimum, and maximum number of pending SCSI commands for writes and reads	An average close to the maximum (32) might indicate saturation of zfcps internal resources.
Number of error recovery actions for LUN	Increase might indicate hardware problems.
Number of LUN resets and target resets for LUN	Increase might indicate hardware problems.

Figure 5-40 shows a sample output of the z/VM Performance Toolkit.

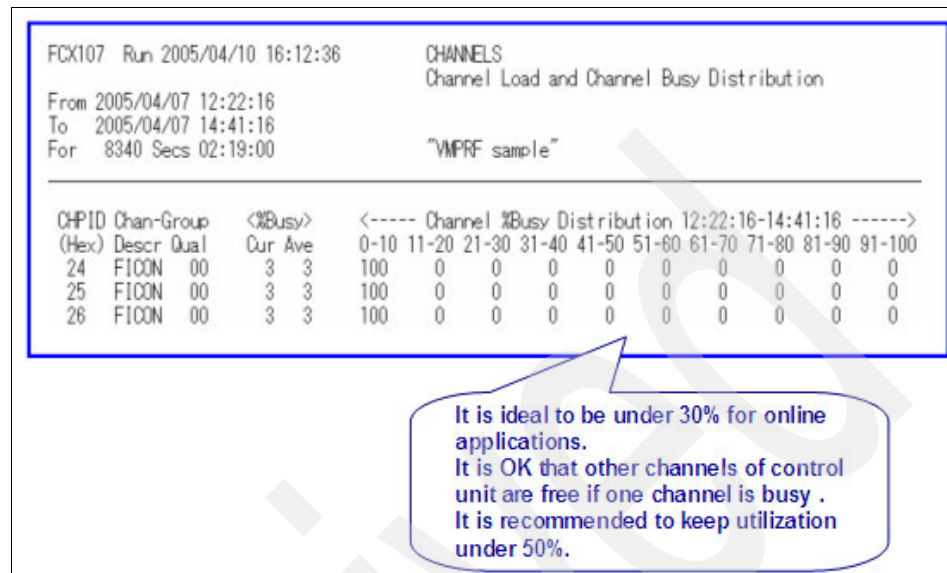


Figure 5-40 Investigating channel bottleneck by using the z/VM Performance Toolkit

This output indicates the channel load and channel busy distribution. As noted, we recommend keeping the channel utilization below 50% for good performance. If this value is above 50%, then it may make sense to investigate deeper for performance problem determination.

Investigating for bottlenecks at the storage subsystem level depends on the type of storage subsystem used and many other factors. Typically, an investigation is necessary at the RAID rank level, DA pair level, HBA level, and so on. The cache and processor used in the storage subsystem may also show a bottleneck at times depending on the load that it is carrying. In the case of sequential I/O, interest should be in measuring the throughput, and in the case of random I/O, the interest should be in measuring the IOPS and response time. There are tools available to assist in determining the bottleneck at the storage subsystem level such as the IBM TotalStorage® Productivity Center for Disk. A more detailed explanation of storage subsystem level analysis is beyond the scope of this book. Refer to the documentation for the storage subsystem used at your site.

Network I/O bottlenecks

The performance of the network subsystem depends on other resources on the system such as CPU power, disk performance, applications used, and so on. For example, HiperSockets operate the real memory speed, but when the data is

written on to the disk, the speed of the disk is what really becomes the bottleneck.

Figure 5-41 and Figure 5-42 on page 203 show the different steps involved in the problem determination of network I/O performance. The type of the networking option used (such as OSA Express, HiperSockets, VSWITCH/Guest LAN), the way these networking options are configured (for example, Layer 2 or Layer 3, Channel Bonding, MTU size, and so on), and the network environment to which the system is connected are the main things to be noted while checking the current configuration.

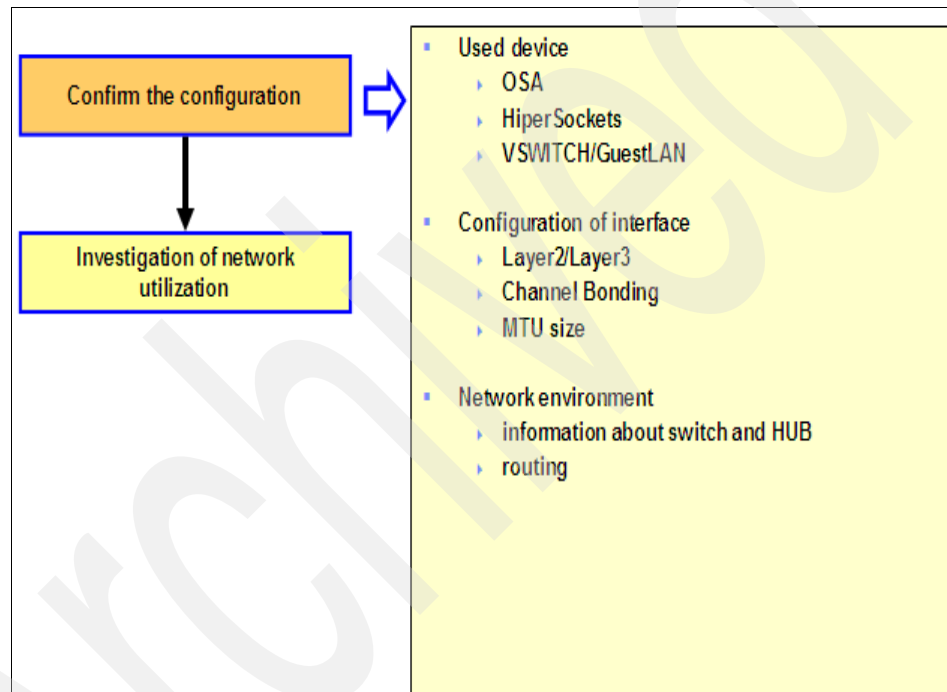


Figure 5-41 Network performance problem determination flow

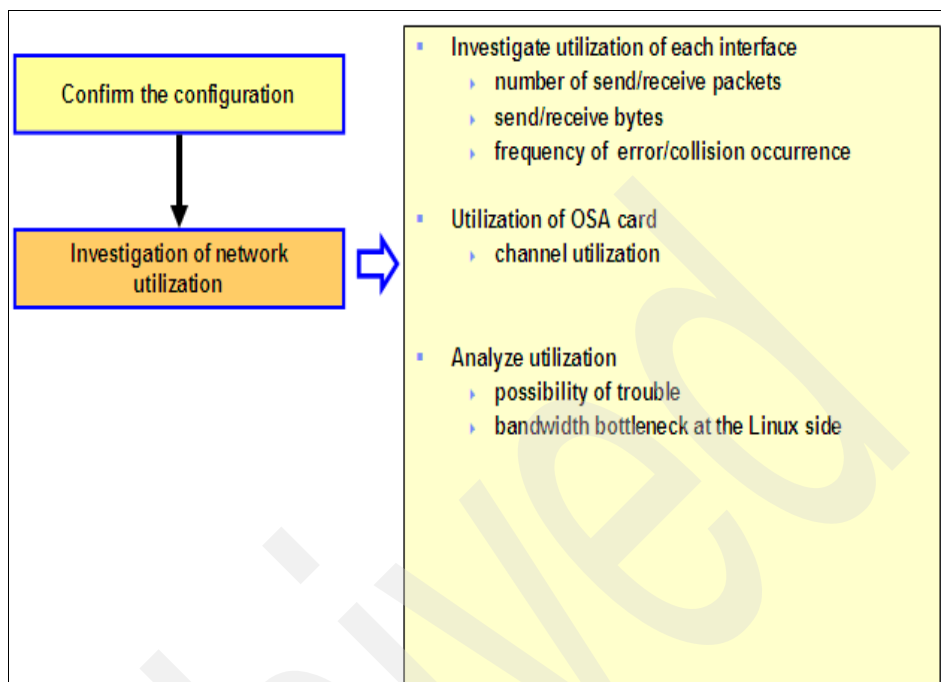


Figure 5-42 Network performance problem determination - investigating utilization

When the network utilization is being investigated, the utilization of each interface needs to be investigated (the number of packets transmitted and received, frequency of the occurrence of errors and collisions, and so on, to be analyzed). If there are many collision packets, then there is a possibility of hardware trouble or a possibility of trouble in the network environment. In the case of an OSA card, the channel utilization must be examined. Investigate whether the bandwidth of OSA or HiperSockets becomes full by referring performance reports. At the Linux side check the bandwidth bottleneck, incorrect configurations, and so on.

Note: The HiperSockets performance report can be found at:

<http://www.vm.ibm.com/perf/reports/zvm/html/iqdio.html>

The OSA Express performance report can be found at:

<http://www.vm.ibm.com/perf/reports/zvm/html/layer2.html>

If the bandwidth is full, it should be examined as to whether it can be resolved by fine-tuning configuration parameters, by using channel bonding for load balancing, optimizing the applications to reduce the packets transmitted by utilizing compression, and so on.

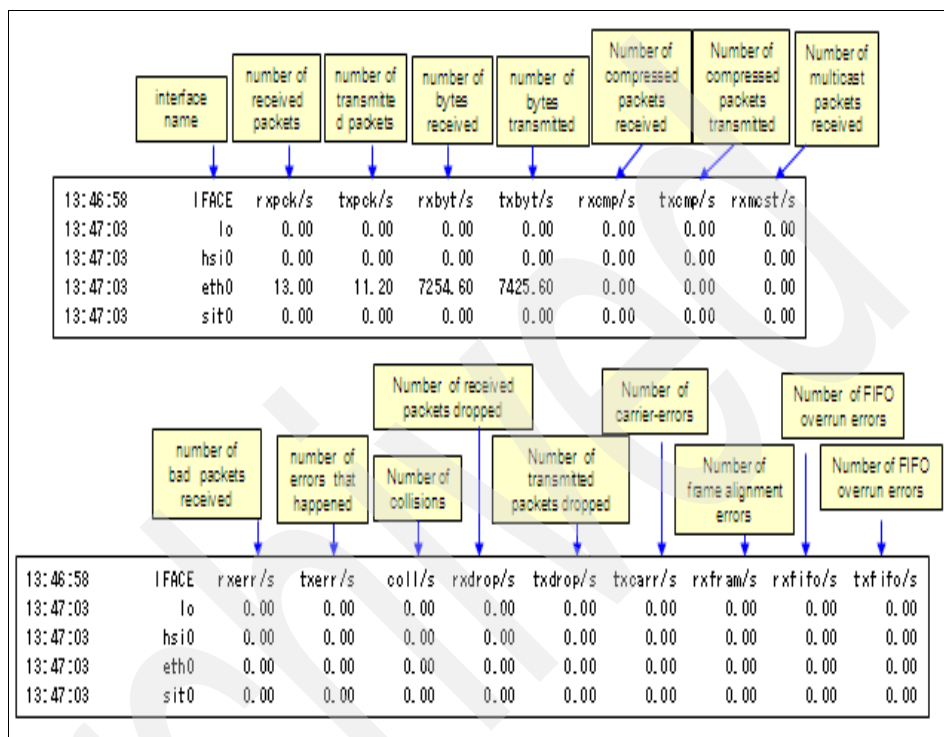


Figure 5-43 Network utilization check by using sar

Data for analysis

To troubleshoot the problem, we need to gather and analyze the following information:

- ▶ sar output
- ▶ vmstat output
- ▶ iostat output

Analysis and conclusion

Figure 5-45 shows the graph prepared from the vmstat output. From this graph, it is apparent that there are times at which all the CPUs are not being utilized. This is not because the other processes are utilizing the CPU but because of the application limitation.

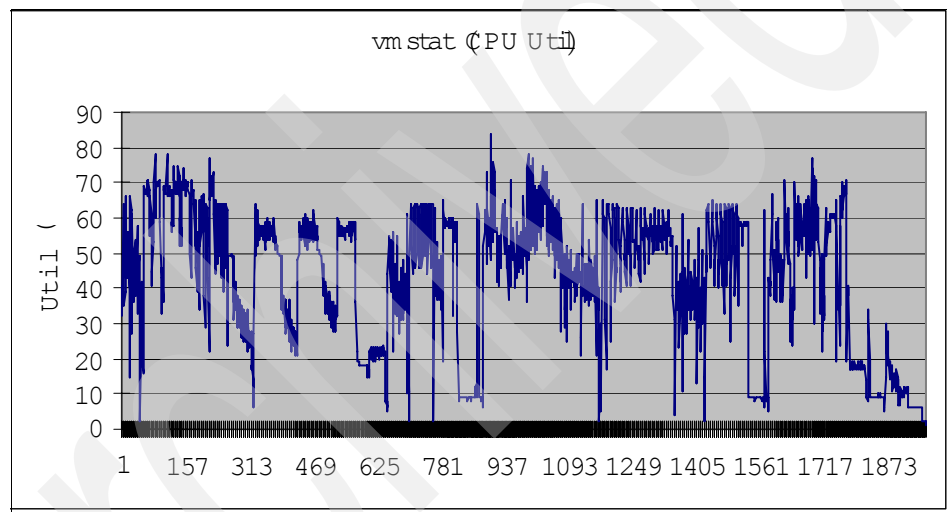


Figure 5-45 CPU utilization report based on the vmstat output

Figure 5-46 shows the graph plotted using the “Run Queue” data from the vmstat output. This system has 16 CPUs, so at any point in time at least 16 processes can run. But an examination of the graph shows that there are times at which just about 10 processes were running and all the CPUs are not utilized effectively. So, the batch job can be reconfigured to utilize all the CPUs effectively to shorten the run time. The CPU utilization can be improved (close to 100%), too.

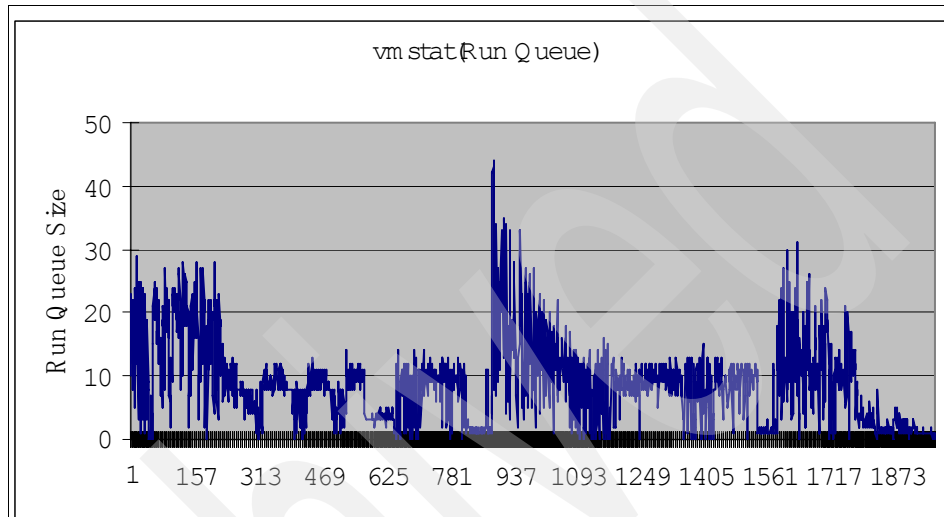


Figure 5-46 Graph based on the vmstat Run Queue data

Figure 5-47 shows an example of improved CPU utilization.

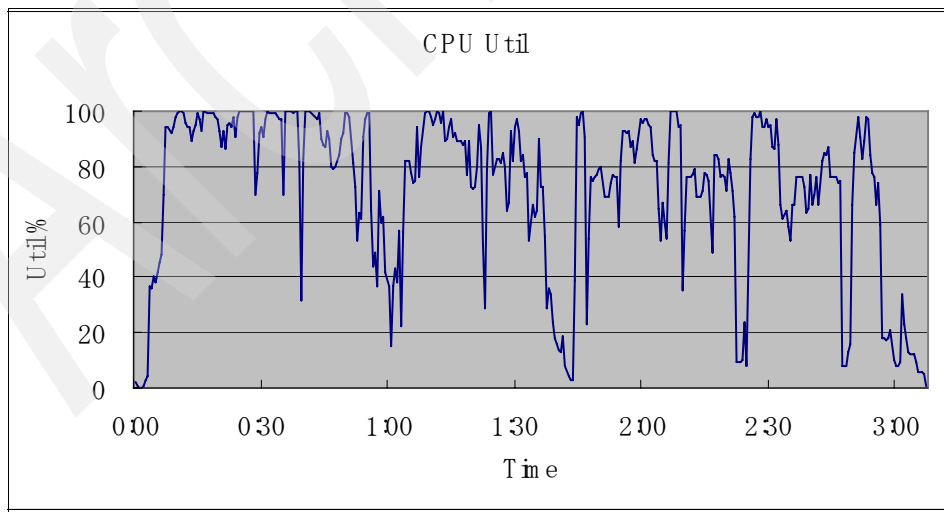


Figure 5-47 Improved CPU utilization

This is a classic case in which we experience that a performance bottleneck can arise even when the all the resources on the system are not used.

Case study 2 - slow database server response

In this case study we examine a problem reported as a slow database server response.

Problem

Performance tests done on a database server indicate that the expected results are not attained.

Data for analysis

To troubleshoot the problem, we need to gather and analyze the following information:

- ▶ sar, vmstat, and iostat outputs
- ▶ Information about the disk configuration
- ▶ Oracle statspack output

Analysis and conclusion

The database used is Oracle. From the Oracle statspack output shown in Figure 5-48 it is clear that the *log file sync* event is taking a long time to complete. *log file sync* is the event that writes the updated data in the log buffer into online redo log files when dirty blocks in the dirty buffer are written onto the disks.

Top 5 Timed Events ~~~~~				
Event	Waits	Time (s)	Avg wait (ms)	%Total Call Time
log file sync	960,319	1,574	2	48.7
cr request retry	900	836	929	25.9
log file parallel write	514,887	316	1	9.8
CPU time		237		7.4
gc current block busy	7,575	94	12	2.9

Figure 5-48 Oracle statspack output

Upon investigation of the disk volume hosting the redo log file, and the disk I/O for that volume, it became clear that logical volume is not striped. Because of this, all the I/O was concentrated on one disk rank and there was a bottleneck. So, striping the volume to multiple ranks solved the issue, since that resulted in distributing the I/O evenly across multiple disk ranks instead of concentrating only on one. This is an example of a performance bottleneck because of an incorrect system configuration.

Case study 3 - using OProfile for problem determination

In this case study we demonstrate problem determination by using OProfile.

Problem

The porting of an ISV middleware and application is completed. While running a test with peak load, it was noticed that even the target response is not achieved and the CPU utilization is almost 100%. This was beyond the estimated CPU requirement for the application.

Data for analysis

To troubleshoot the problem, we need to gather and analyze the following information:

- ▶ sar and vmstat outputs
- ▶ Data gathered using OProfile

Analysis and conclusion

From the CPU utilization report generated using vmstat, it was found that the utilization of CPU under peak load is usr: 75%, sys: 20%, and iowait: 5%. Since most of the CPU utilization is occurring for the usr part with lot of I/O, the next logical thing is to check the middleware or application. The CPU utilization data gathered by using OProfile is shown in Figure 5-49. So, the function aaaaaaaa in the library AAAAAA.so.1 provided by the middleware is called frequently. This was discussed with the ISV and a patch for the middleware fixed the problem.

CPU: CPU with timer interrupt, speed 0 MHz (estimated)			
Profiling through timer interrupt			
samples	%	app name	symbol name
168856	37.6918	AAAAAA.so.1	aaaaaaa
17815	3.9766	vmlinux	default_idle
15248	3.4036	vmlinux	_spin_unlock_irq
13024	2.9072	BBBBBBBB	bbbbbbbb
7354	1.6416	ext3	(no symbols)
5587	1.2471	libc-2.3.4.so	memset
5457	1.2181	BBBBBBBB	ccccccc
5307	1.1846	CCCCCCCC	(no symbols)
5145	1.1485	vmlinux	number
5018	1.1201	libc-2.3.4.so	_IO_vfscanf

Figure 5-49 CPU utilization captured by using OProfile

This example illustrated how to troubleshoot a performance bottleneck using OProfile in conjunction with other tools. Sometimes it is necessary to interlock with the middleware or application provider to resolve the problem.

Storage problems

This chapter discusses storage problem determination for Linux on System z. Like the other parts of this book, the focus of this chapter is on Linux running on z/VM.

In this chapter, we discuss:

- ▶ How to determine a storage-related problem
- ▶ Case study

6.1 How to determine a storage-related problem

To determine whether the problem that you encounter on your Linux system is related to storage, you need to know commands to use to investigate the storage devices. In this section we discuss the different commands to use for the different types of storage devices.

For an overview of your storage devices use the **lscss** command. **lscss** is used to list all or a subset of devices that are managed by the common I/O subsystem. It reads the status from **sysfs** and lists all channel path (CHP) information, whether it is online or not. The status of the channel subsystem can also be checked.

In Figure 6-1 you can see the **lscss** command with some explanation.

# lscss										
Device	Subchan.	DevType	CU	Type	Use	PIM	PAM	POM	CHPIDs	
0.0. C65D	0.0. 000A	3390/0A	3990/E9	yes	F0	F0	FF	C0C2C1C3	00000000	
0.0. 38C6	0.0. 0010	1732/03	1731/03	yes	80	80	FF	B6000000	00000000	
0.0. 39C6	0.0. 0011	1732/03	1731/03	yes	80	80	FF	B7000000	00000000	

Figure 6-1 Increased I/O rate

If a device is not listed with the **lscss** command, check the following:

- ▶ Is the IOCP definition correct?
- ▶ Is the target device correct defined in z/VM?
- ▶ Check the **cio_ignore** parameter.

If the address of the target device is listed with the **lscss** command but the Use column is blank, depending on the way the devices are brought online, check:

- ▶ Via kernel parameter in **zipl.conf**
 - a. Check the **zipl.conf** file.
 - b. After editing the **zipl.conf**, run the **zipl** command.

- ▶ Via initrd
 - a. Check initrd, then run **zipl**.
 - b. Check the linuxrc script in initrd (to verify the procedures).
- ▶ Via coldplug
 - Check /etc/sysconfig/hardware/hwcfg-*.

If the channel status is not normal, check the cabling, HMC, and so on.

cio_ignore

Linux senses all devices defined for the LPAR. All detected devices are controlled under **sysfs** and can be varied online or offline with Linux commands.

cio_ignore is a kernel parameter that can filter the target device address detected by Linux. The devices set on **cio_ignore** are not detected and not controlled under **sysfs** and cannot be varied online or offline with a Linux command. These devices are also not displayed with the **lscss** command.

6.1.1 How to check DASD devices

To check the status of the DASD devices use the **lsdasd** command. With this command you get some basic information about your DASD devices. In Figure 6-2 you see the **lsdasd** command with the response from the system and some explanation of how to interpret the system messages.

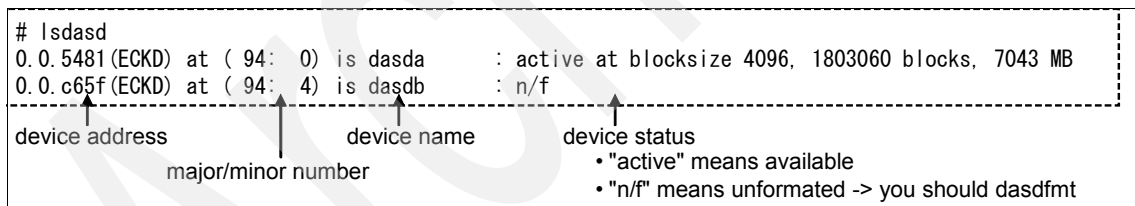


Figure 6-2 *lsdasd* command

The **dasdview** command gives you more information about the DASD and the volume table of contents (VTOC). This command has many options to choose to display the information you need:

- b** Prints a DASD dump from 'begin'
- s** Prints a DASD dump with size 'size'
- 1** Uses format 1 for the dump (default)

- 2 Uses format 2 for the dump
 'begin' and 'size' can have the following format:
 x[klmlbltlc]
 for x byte, kilobyte, megabyte, blocks, tracks, and
 cylinders
- i Prints general DASD information and geometry
- x Prints extended DASD information
- j Prints the volume serial number (volume identifier)
- l Prints information about the volume label
- t Prints the table of contents (VTOC)
- h Prints this usage text
- v Prints the version number
- n Specifies the device by a device number (needs devfs)
- f Specifies a device by a device node

Two examples of the **dasdview** command are shown in Figure 6-3 and Figure 6-4 on page 215.

```
# dasdview -i /dev/dasda
--- general DASD information ---
device node      : /dev/dasda
busid            : 0.0.9144
type             : ECKD
device type      : hex 3390          dec 13200
--- DASD geometry ---
number of cylinders : hex d0b          dec 3339
tracks per cylinder : hex f            dec 15
blocks per track    : hex c            dec 12
blocksize          : hex 1000         dec 4096
```

Figure 6-3 *dasdview* command with the -i option

```
# dasdview -x /dev/dasda
:
--- extended DASD information -----
real device number      : hex 5          dec 5
subchannel identifier   : hex b          dec 11
CU type (SenseID)      : hex 3990       dec 14736
CU model (SenseID)     : hex e9         dec 233
device type (SenseID)  : hex 3390       dec 13200
device model (SenseID) : hex a          dec 10
open count              : hex 3          dec 3
:
:
```

Figure 6-4 *dasdview* command with the *-x* option

With the **fdasd** command you can check the partition table of the DASD. An example is shown in Figure 6-5.

```
# fdasd -p /dev/dasda
:
----- tracks -----
      Device      start    end    length    ld  System
    /dev/dasda1         2    2744     2743     1  Linux native
    /dev/dasda2    2745   50084    47340     2  Linux native
exiting...
```

Figure 6-5 *fdasd* command

Notice the existence of the device nodes (/dev/dasda1 and /dev/dasda2). They are created by udev /dynamic device management.

6.1.2 How to check FCP/SCSI devices

If you are using SCSI devices in your Linux system you can use **lsscsi** to check the status of these devices. This command reads the status from sysfs and displays all SCSI devices that are online.

Figure 6-6 on page 216 shows a sample of the output with some explanation of the **lsscsi** command.

```
# lsscsi -v
sysfsroot: /sys
[0:0:1:0]   disk   IBM      2105800      3. 79  /dev/sda
dir: /sys/bus/scsi/devices/0:0:1:0  [/sys/devices/css0/0.0.0010/0.0.38c6/host0/0:0:1:0]
[0:0:1:1]   disk   IBM      2105800      3. 79  /dev/sdb
dir: /sys/bus/scsi/devices/0:0:1:1  [/sys/devices/css0/0.0.0010/0.0.38c6/host0/0:0:1:1]
[1:0:1:0]   disk   IBM      2105800      3. 79  /dev/sdc
dir: /sys/bus/scsi/devices/1:0:1:0  [/sys/devices/css0/0.0.0011/0.0.39c6/host1/1:0:1:0]
[1:0:1:1]   disk   IBM      2105800      3. 79  /dev/sdd
dir: /sys/bus/scsi/devices/1:0:1:1  [/sys/devices/css0/0.0.0011/0.0.39c6/host1/1:0:1:1]
```

Figure 6-6 **lsscsi** command

If the target address is not displayed with **lsscsi**, you may want to check the following, depending on the way the disk was brought online:

- ▶ With **initrd**
 - a. After modifying **initrd**, run **zipl**.
 - b. Check the **linuxrc** script in **initrd** (to verify the procedures).
- ▶ **coldplug**
Check **/etc/sysconfig/hardware/hwcfg-***, especially, **WWPN** and **LUN ID**.

Check whether the target LUNs are mapped. Use **san_disc** to verify:

- ▶ SAN switch zoning
- ▶ LUN mapping
- ▶ Cabling

With the command **lszfc** you can get more information about zfc adapters, ports, and units that are online. In Figure 6-7 you can see a example of a **lszfc** command.

```
# lszfc -H -D -P
0.0.3ec4 host0 }
0.0.3fc4 host1 }
0.0.3ec4/0x5005076303008455 rport-0:0-0 }
0.0.3fc4/0x500507630300c455 rport-1:0-0 }
0.0.3ec4/0x5005076303008455/0x4051402100000000 0:0:0:0 }
0.0.3ec4/0x5005076303008455/0x4050402200000000 0:0:0:1 }
0.0.3fc4/0x500507630300c455/0x4050402200000000 1:0:0:0 }
0.0.3fc4/0x500507630300c455/0x4051402100000000 1:0:0:1 }
```

-H : list information about HBA
-P : list information about ports
-D : list information about units

Figure 6-7 lszfc command

With the SAN discovery tool and using command **san_disc**, you can display information about SANs such as the world wide port name (wwpn) of ports on storage and the logical unit identifier (lunid).

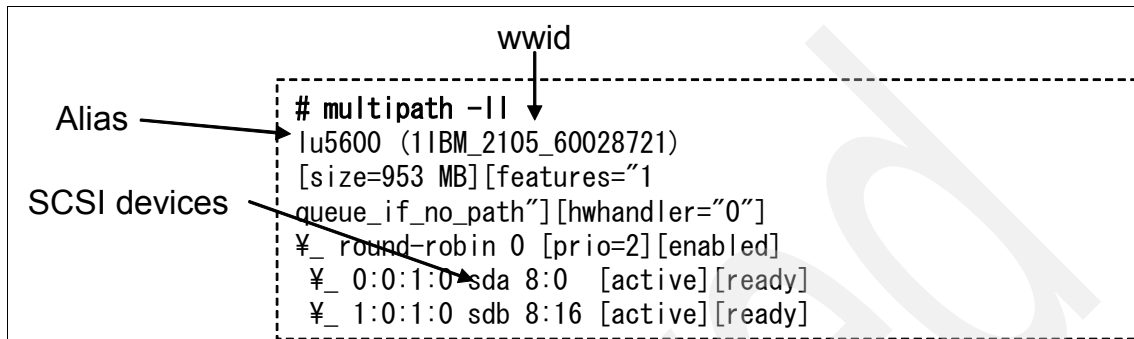
This is only available on SLES9 SP3 and SLES10. You need to load the **zfc_hbaapi** module. See an example of the **san_disc** command in Figure 6-8.

```
# san_disc -c HBA_LIST
Number of Adapters: 2
No. Port WWN Node WWN SerialNumber Busid
1 0x500507640122038d 0x5005076400c4e24d IBM5100000004E24D 0.0.38c6
2 0x50050764012203b3 0x5005076400c4e24d IBM5100000004E24D 0.0.39c6
# san_disc -c PORT_LIST -a 1
No. Port WWN Node WWN DID Type
1 0x500507640122038d 0x5005076400c4e24d 0x030a00 N_Port
2 0x5005076300c1adc9 0x5005076300c0adc9 0x030c00 N_Port
3 0x500507640122038d 0x5005076400c4e24d 0x030a00 N_Port
# san_disc -c REPORT_LUNS -a 1 -p 0x5005076300c1adc9
Number of LUNs: 4
No. LUN
1 0x5200000000000000
2 0x5201000000000000
3 0x5202000000000000
4 0x5203000000000000
```

Figure 6-8 san_disc command

6.1.3 What to check for multipathing

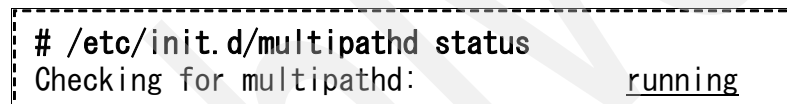
The command `multipath -ll` reads the sysfs and displays the multipath devices (Figure 6-9).



```
# multipath -ll
lu5600 (1IBM_2105_60028721)
[size=953 MB][features="1
queue_if_no_path"][hwhandler="0"]
¥_ round-robin 0 [prio=2][enabled]
¥_ 0:0:1:0 sda 8:0 [active][ready]
¥_ 1:0:1:0 sdb 8:16 [active][ready]
```

Figure 6-9 `multipath -ll` command

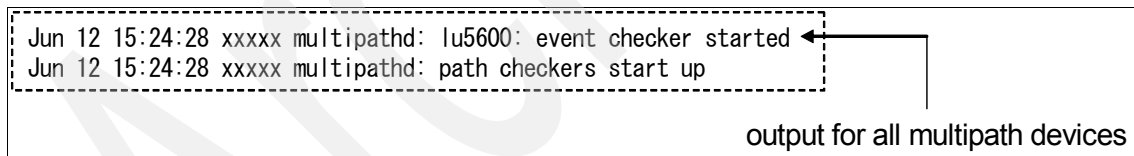
The `/etc/init.d/multipathd` status checks the status of `multipathd` (Figure 6-10).



```
# /etc/init.d/multipathd status
Checking for multipathd:      running
```

Figure 6-10 `/etc/init.d/multipathd`

The path checker's log is stored to `/var/log/messages` (Figure 6-11).



```
Jun 12 15:24:28 xxxxx multipathd: lu5600: event checker started
Jun 12 15:24:28 xxxxx multipathd: path checkers start up
```

output for all multipath devices

Figure 6-11 Path checker

in the case that the target disk is displayed with `multipath -ll`, the disk must be varied online in `initrd` for multipathing on booting.

If multipathing can be configured with the `multipath` command after booting, disks may be brought online in `coldplug`:

1. Check the `linuxrc` script in `initrd`.
2. After modifying `initrd`, run `zipl`.

Verify multipath.conf. devnode_blacklist is a parameter to filter the target devices of multipathing. If alias is invalid, wwid is used as the device name.

When configuring multipathing, the files named alias are created:

- ▶ /dev/mapper/
- ▶ /dev/disk/by-name/

Check the partition tables in boot.multipath:

- ▶ Configure multipath.
- ▶ Read partition table.
- ▶ dev/mapper/aliaspN (N is partition number) is created.

See an example of the /dev/mapper in Figure 6-12.

```
# ls -l /dev/mapper/
total 118
drwxr-xr-x  2 root root   408 May  1 10:55 .
drwxr-xr-x 23 root root 121088 May  1 10:55 ..
crw----- 1 root root  10, 63 May  1 10:54 control
brw----- 1 root root 253,  1 May  1 10:54 lu5201 ← alias
brw----- 1 root root 253,  3 May  1 10:54 lu5201p1 ←
brw----- 1 root root 253,  2 May  1 10:54 lu5202 ←
brw----- 1 root root 253,  4 May  1 10:54 lu5202p1 ← partition
brw----- 1 root root 253,  5 May  1 10:55 vgA-lvA
brw----- 1 root root 253,  6 May  1 10:55 vgB-lvB
brw----- 1 root root 253,  0 Jan 25 21:39 vgroot-lvroot
```

(p1 means the first partition)

Figure 6-12 /dev/mapper

6.1.4 What to check for LVM

When you are using LVM in your Linux system, here are some things you should check in case of problems within the LVM

Check LVM recognition

With the command `vgdisplay -v` you get some useful information about the LVM volume group. An example of the `vgdisplay -v` command is shown in Figure 6-13 and continued in Figure 6-14 on page 221.

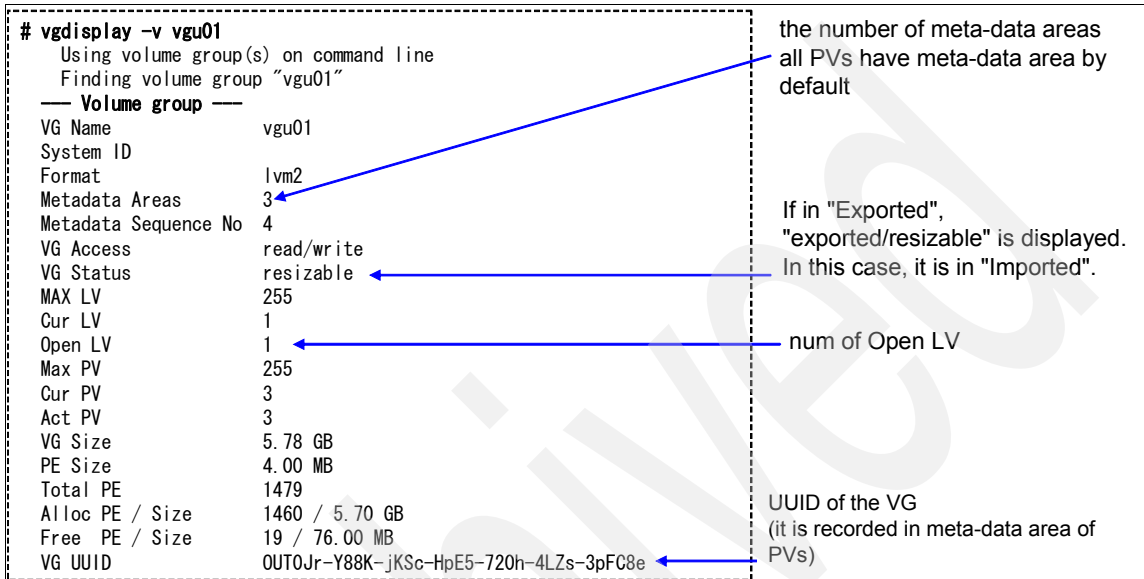


Figure 6-13 `vgdisplay -v`

: (continued)		
— Logical volume —		LVs in the VG
LV Name	/dev/vgu01/lvu01	
VG Name	vgu01	
LV UUID	REw07H-erkC-Z3P5-yzFb-1aoW-HTLy-	UUID of the LV (it is recorded in meta-data area of PVs)
c0aMPR		
LV Write Access	read/write	
LV Status	available	If the vg id deactivated, "NOT available" is displayed.
# open	2	num of processes which open the LV. If it is not "0", the LV cannot be deactivated.
LV Size	5.70 GB	
Current LE	1460	
Segments	3	
Allocation	inherit	
Read ahead sectors	0	
Block device	253:8	device number of the LV (major:minor) These have same major:minor number. • /dev/mapper/vgu01-lvu01 • /dev/dm-8 • /dev/vgu01/lvu01
— Physical volumes —		PVs which compose the VG.
PV Name	/dev/dasdc1	
PV UUID	5pXxDK-9M1Z-zH4i-iyMQ-Jny6-YtCW-	UUID of the PV (it is recorded in meta-data area of PVs)
8GzIK0		
PV Status	allocatable	
Total PE / Free PE	307 / 0	
:		

Figure 6-14 `vgdisplay -v` (continued)

With the **vgscan** command, all scanned devices are stored in the `./etc/lvm/.cache` file. The target device file name is defined by filter parameter in `lvm.conf`.

```
persistent_filter_cache {
    valid_devices=[
        "/dev/dm-6",
        "/dev/dasdc",
        "/dev/dasdb1",
        "/dev/dasdc1",
        "/dev/dasda1",
        "/dev/dm-5",
        "/dev/mapper/lu5601p1",
        :
        :
    ]
}
```

Figure 6-15 `lvm.conf` file

The recognition of the LVM by the **vgscan** command is the following sequence:
PVs → VGs → LVs.

Unless all PVs that are part of a VG are recognized, the VG is not available. First PVs need to be recognized in `vgscan` while booting. PVs need to be recognized in `initrd` or `zipl.conf`.

The **pvdisplay** command is available to display the recognized PVs.

In the case that PVs are recognized but the VG is not recognized, the meta-data area in a PV may be broken.

In the case that PVs are not recognized (not listed by `pvdisplay`), check the following:

- ▶ In `lvm.conf`:
 - filter parameter
 - types parameter
- ▶ Check `./etc/lvm/.cache` to verify the target device file name.

6.1.5 I/O statistics

Another source of information about the storage devices is the DASD I/O statistics. With I/O statistics all the important information about the storage devices are collected and stored in a file for further analysis. You can switch the DASD I/O statistics on and off as needed. Figure 6-16 shows how to switch DASD I/O statistic on and off.



```
9.12.4.185 - PuTTY
lnxdb2:/ # echo set on > /proc/dasd/statistics
lnxdb2:/ # echo set off > /proc/dasd/statistics
lnxdb2:/ #
```

Figure 6-16 Switch DASD I/O statistic on/off

To reset the collection, switch DASD I/O statistics off and on again.

Also see 3.1.8, “netstat” on page 68, for more informational about DASD I/O statistics.

6.2 Case study

In this section we see a case study involving a real-life problem. In this case study, we discuss a scenario in which the swap rate becomes too high and the system throughput decreases dramatically.

We have a Linux guest system that runs a DB2 database. The Linux system has an increase its workload due to DB2 requests. We used z/VM tools and Linux basic tools to monitor this situation. In the first page (Figure 6-17) of the performance toolkit window FCX100 the load and the paging activity on the system is normal.

FCX100	CPU 2094	SER 2991E	Interval 11:02:21 - 11:03:21						Perf. Monitor				
CPU Load										Vector Facility		Status or	
PROC	TYPE	%CPU	%CP	%EMU	%WT	%SYS	%SP	%SIC	%LOGLD	%VTOT	%VEMU	REST	ded. User
P00	CP	10	1	8	90	1	0	96	10	Master
P01	CP	10	1	9	90	0	0	96	10	Alternate
P02	CP	8	1	7	92	0	0	96	8	Alternate
P03	CP	8	1	7	92	0	0	96	8	Alternate
Total SSCH/RSCH				13/s		Page rate			6.5/s	Priv. instruct.			2266/s
Virtual I/O rate				14/s		XSTORE paging			3.7/s	Diagnose instr.			3/s
Total rel. SHARE				500		Tot. abs SHARE			0%				
Queue Statistics:				Q0	Q1	Q2	Q3	User Status:					
VMDBKs in queue				0	1	0	6	# of logged on users					24
VMDBKs loading				0	0	0	0	# of dialed users					0
Eligible VMDBKs					0	0	0	# of active users					13
El. VMDBKs loading					0	0	0	# of in-queue users					7
Tot. WS (pages)				0	669	0	820955	% in-Q users in PGWAIT					0
Expansion factor					0	0	0	% in-Q users in IOWAIT					1
85% elapsed time				1.376	.172	1.376	8.256	% elig. (resource wait)					0
Command ==>													
F1=Help F4=Top F5=Bot F7=Bkwd F8=Fwd F12=Return													

Figure 6-17 IBM z/VM Performance Toolkit shows normal load and paging activity


```

FCX115          CPU 2094  SER 2991E  Interval 10:08:37 - 10:08:38  Perf. Monitor

Detailed data for user LNXDB2
Total CPU      : 2.5%      Storage def. : 2048MB      Page fault rate: .0/s
Superv. CPU    : .3%      Resident <2GB: 79391      Page read rate : .0/s
Emulat. CPU    : 2.2%     Resident >2GB: 37211      Page write rate: .0/s
VF total       : ....%    Proj. WSET   : 168614      Pgs moved >2GB>: .0/s
VF overhead    : ....%    Reserved pgs : 0          Main > XSTORE  : .0/s
VF emulation   : ....%    Locked pages  : 5          XSTORE > main  : .0/s
VF load rate   : ....%    XSTORE dedic.: 0MB        XSTORE > DASD  : .0/s
I/O rate       : 222/s     XSTORE pages  : 70160      SP00L pg reads : .0/s
DASD IO rate   : .0/s     DASD slots    : 52619      SP00L pg writes: .0/s
UR I/O rate    : .0/s     IUCV X-fer/s  : .0/s        MDC insert rate: .0/s
Diag. X'98'    : .0/s     Share         : 100       MDC I/O avoided: .0/s
*BLOCKIO       : .0/s     Max. share    : ...

#I/O active : 0      Active :100%    PSW wait : 97%    I/O act. : 0%
Stacked blk : ..     Page wait : 0%    CF wait  : 0%     Eligible : 0%
Stat.: EME,P02,PSWT I/O wait : 0%    Sim. wait: 1%     Runnable : 1%

Proc.  %CPU  %CP  %EM  %VECT %VOHD %VEMU VLD/S  IO/S  Status
00      2.5   .3  2.2   ...   ...   ...   ...   221  EME,P02,PSWT
01      .0   .0  .0   ...   ...   ...   ...   .0   EME,P02,PSWT

Data Space Name      Size Mode  PgRd/s PgWr/s XRd/s XWr/s Migr/s Steal/s
BASE                 2048MB Priv    .0    .0    .0    .0    .0    .0

Device activity and status:
0009 3215 .0          000C 254R      CL *, EOF    NOH NCNT

Command ==>
F1=Help  F4=Top  F5=Bot  F7=Bkwd  F8=Fwd  F12=Return

```

In Figure 6-19 (an image of the overall system performance window) we see that the XSTORE paging is already at nearly 122 per second. This indicates that there is heavy paging activity.

FCX100	CPU 2094	SER 2991E	Interval 10:07:21 - 10:08:21							Perf. Monitor				
CPU Load											Vector Facility		Status or	
PROC	TYPE	%CPU	%CP	%EMU	%WT	%SYS	%SP	%SIC	%LOGLD	%VTOT	%VEMU	REST	ded.	User
P00	CP	4	1	3	96	1	0	95	4		Master
P01	CP	4	1	3	96	0	0	94	4		Alternate
P02	CP	4	1	4	96	0	0	94	4		Alternate
P03	CP	4	1	4	96	0	0	94	4		Alternate
Total SSCH/RSCH				8/s		Page rate				14.4/s		Priv. instruct.		891/s
Virtual I/O rate				4/s		XSTORE paging				121.8/s		Diagnose instr.		6/s
Total rel. SHARE				400		Tot. abs SHARE				0%				
Queue Statistics:				Q0	Q1	Q2	Q3	User Status:						
VMDBKs in queue				0	0	0	6	# of logged on users						24
VMDBKs loading				0	0	0	0	# of dialed users						0
Eligible VMDBKs					0	0	0	# of active users						12
El. VMDBKs loading					0	0	0	# of in-queue users						6
Tot. WS (pages)				0	0	0	1007k	% in-Q users in PGWAIT						0
Expansion factor					0	0	0	% in-Q users in IOWAIT						0
85% elapsed time				1.656	.207	1.656	9.936	% elig. (resource wait)						0
Transactions				Q-Disp	trivial	non-trv	User Extremes:							
Average users				.0	.0	.1	Max. CPU %		LNHWAS	6.4				
Trans. per sec.				.8	1.7	.5	Max. VECT %					
Av. time (sec)				.101	.017	.327	Max. IO/sec		LNHWB2	1.5				
UP trans. time					.017	.327	Max. PGS/s		LNHWAS	8.6				
MP trans. time					.000	.000	Max. RESPG		LNHWAS	711653				
System ITR (trans. per sec. tot. CPU)						20.2	Max. MDCIO					
Command ==>														
F1=Help F4=Top F5=Bot F7=Bkwd F8=Fwd F12=Return														

Figure 6-19 Heavy paging activity

After a couple of minutes we see that the I/O rate of the LNXDB2 Linux guest system increased to 583 per second.

FCX115										CPU 2094		SER 2991E		Interval 10:12:27 - 10:12:28				Perf. Monitor	
Detailed data for user LNXDB2																			
Total CPU		: 14.3%		Storage def.		: 2048MB		Page fault rate:		: .9/s									
Superv. CPU		: 1.4%		Resident <2GB:		77876		Page read rate :		: .9/s									
Emulat. CPU		: 12.9%		Resident >2GB:		33658		Page write rate:		: .0/s									
VF total		:%		Proj. WSET		: 168614		Pgs moved >2GB>:		: .0/s									
VF overhead		:%		Reserved pgs		: 0		Main > XSTORE		: .0/s									
VF emulation:		:%		Locked pages		: 5		XSTORE > main		: 6.4/s									
VF load rate:		:s		XSTORE dedic.:		OMB		XSTORE > DASD		: .0/s									
I/O rate		: 583/s		XSTORE pages		: 75125		SPOOL pg reads		: .0/s									
DASD IO rate:		: .0/s		DASD slots		: 52754		SPOOL pg writes:		: .0/s									
UR I/O rate		: .0/s		IUCV X-fer/s		: .0/s		MDC insert rate:		: .0/s									
Diag. X'98'		: .0/s		Share		: 100		MDC I/O avoided:		: .0/s									
*BLOCKIO		: .0/s		Max. share		: ...													
#I/O active		: 0		Active		:100%		PSW wait		: 97%		I/O act.		: 0%					
Stacked blk		: ..		Page wait		: 0%		CF wait		: 0%		Eligible		: 0%					
Stat.: EME,P02,PSWT				I/O wait		: 0%		Sim. wait:		: 1%		Runnable		: 2%					
Proc.		%CPU		%CP		%EM		%VECT		%VOHD		%VEMU		VLD/S		IO/S		Status	
00		13.6		1.4		12.2			577		EME,P02,PSWT	
01		.7		.0		.7			5.5		EME,P02,PSWT	
Data Space Name				Size		Mode		PgRd/s		PgWr/s		XRd/s		XWr/s		Migr/s		Steal/s	
BASE				2048MB		Priv		.9		.0		6.4		.0		.0		.0	
Device activity and status:																			
0009 3215		.0						000C 254R				CL *, EOF				NOH NCNT			
Command ==>																			
F1=Help		F4=Top		F5=Bot		F7=Bkwd		F8=Fwd		F12=Return									

Table 6-1 Increase

FCX103	CPU 2094	SER 2991E	Interval 10:11:21 - 10:12:21	Perf. Monitor
Total DPA size	4'013MB	MDCACHE utilization:		
Locked pages	14'984kB	Min. size in XSTORE		0kB
Trace tablekB	Max. size in XSTORE		1'024MB
Pageable	3'999MB	Ideal size in XSTORE		314'688kB
Storage utilization	103%	Act. size in XSTORE		274'304kB
Tasks waiting for a frame	0	Bias for XSTORE		1.00
Tasks waiting for a page	1/s	Min. size in main stor.		0kB
		Max. size in main stor.		4'096MB
V=R area:		Ideal size in main stor.		277'676kB
Size defined	...kB	Act. size in main stor.		176'692kB
FREE storage	...kB	Bias for main stor.		1.00
V=R recovery area in use	...%	MDCACHE limit / user		79'504kB
V=R user	Users with MDCACHE inserts		2
		MDISK cache read rate		0/s
Paging / spooling activity:		MDISK cache write rate	/s
Page moves <2GB for trans.	0/s	MDISK cache read hit rate		0/s
Fast path page-in rate	137/s	MDISK cache read hit ratio		71%
Long path page-in rate	2/s			
Long path page-out rate	147/s	VDISks:		
Page read rate	12/s	System limit (blocks)		3882k
Page write rate	9/s	User limit (blocks)		1005k
Page read blocking factor	23	Main store page frames		0
Page write blocking factor	...	Expanded stor. pages		0
Migrate-out blocking factor	21	Pages on DASD		38680
Paging SSCH rate	1/s			
SP00L read rate	0/s			
SP00L write rate	0/s			
Command ===>				
F1=Help	F4=Top	F5=Bot	F7=Bkwd	F8=Fwd
			F12=Return	

Table 6-2 XSTORE continues to increase dramatically

Now the XSTORE paging increased dramatically to 1556 per second. This means that the overall system throughput is very low since the system is short on memory, and therefore the paging activity is very high.

FCX100													CPU 2094		SER 2991E		Interval 10:14:21 - 10:15:21				Perf. Monitor	
PROC	TYPE	%CPU	%CP	%EMU	%WT	%SYS	%SP	%SIC	%LOGLD	%VTOT	%VEMU	REST	ded.	User								
P00	CP	9	2	7	91	1	0	95	9		Master								
P01	CP	7	1	6	93	0	0	96	7		Alternate								
P02	CP	7	1	6	93	0	0	96	7		Alternate								
P03	CP	11	1	10	89	0	0	95	11		Alternate								
Total SSCH/RSCH				12/s		Page rate				51.6/s		Priv. instruct.		1856/s								
Virtual I/O rate				7/s		XSTORE paging				1556/s		Diagnose instr.		28/s								
Total rel. SHARE				500		Tot. abs SHARE				5%												
Queue Statistics:				Q0	Q1	Q2	Q3	User Status:														
VMDBKs in queue				1	1	0	6	# of logged on users				24										
VMDBKs loading				0	0	0	0	# of dialed users				0										
Eligible VMDBKs					0	0	0	# of active users				12										
El. VMDBKs loading					0	0	0	# of in-queue users				8										
Tot. WS (pages)				119	530	0	987786	% in-Q users in PGWAIT				0										
Expansion factor					0	0	0	% in-Q users in IOWAIT				0										
85% elapsed time				1.528	.191	1.528	9.168	% elig. (resource wait)				0										
Transactions				Q-Disp	trivial	non-trv	User Extremes:															
Average users				.1	.0	.1	Max. CPU %		LNHWAS	16.9												
Trans. per sec.				1.4	1.9	.5	Max. VECT %													
Av. time (sec)				.088	.014	.222	Max. IO/sec		LNHWAS	1.6												
UP trans. time					.014	.222	Max. PGS/s		LNXSU3	29.7												
MP trans. time					.000	.000	Max. RESPG		LNHWAS	763955												
System ITR (trans. per sec. tot. CPU)						12.9	Max. MDCIO													
Emul. ITR (trans. per sec. emul. CPU)						.0	Max. XSTORE		LNHDB2	95945												
Command ==>																						
F1=Help F4=Top F5=Bot F7=Bkwd F8=Fwd F12=Return																						

Table 6-3 The ninth column from the left shows the emergency scanning rates

As a result of this high workload, shortage of memory, and extremely high paging activity, the system is in a so-called emergency scanning mode. This can be seen in Table 6-4 on page 230. See the ESCN field, which indicates that the system is short on memory and looking for free pages.

Emergency scanning is an indicator that the system is critically short of storage for some reason. When short of storage pages, the system begins a three-level scanning process: scan 1, scan 2, and then the most serious, emergency scanning. The system is being reduced to extreme measures while hunting for main storage pages, and the situation is so severe that it will steal them from any user. Ultimately, whether they are inactive, in queue, or active, it eventually even steals pages from shared segments and the CP system process.

FCX101	CPU 2094	SER 2991E	Interval 07:44:00 - 10:16:21								Perf. Monitor			
TIME >	PPAG	%ST	ALO/S	FPGS	%FR	SHAR	#TW	ESCN	%PGSL	%SPSL	XSTAV	%XS	XAL/S	XAG
09:51	1000k	108	134	...	0	95k	5	77	13	66	1024M	97	228	21
09:52	1000k	18	420	...	0	93k	2	99	14	66	1024M	99	1309	28s
09:53	1000k	30	52	...	0	94k	0	100	14	66	1024M	99	163	43-
09:54	1000k	102	130	...	0	94k	0	94	14	66	1024M	99	420	48R
09:55	1000k	103	223	...	0	94k	0	95	14	66	1024M	99	666	35R
09:56	1000k	103	46	...	0	95k	0	100	14	66	1024M	99	133	53R
09:57	1000k	103	28	...	0	96k	0	80	14	66	1024M	99	82	80R
09:58	1000k	103	15	...	0	96k	0	100	14	66	1024M	99	59	123R
09:59	1000k	103	10	...	0	96k	0	50	14	66	1024M	99	34	188R
10:00	1000k	103	16	...	0	96k	0	100	14	66	1024M	99	49	239R
10:01	1000k	103	25	...	0	96k	0	100	14	66	1024M	99	82	252R
10:02	1000k	103	18	...	0	96k	0	100	15	66	1024M	99	66	252S
10:03	1000k	103	30	...	0	96k	1	57	15	66	1024M	99	101	216S
10:04	1000k	103	12	...	0	96k	0	50	15	66	1024M	99	31	313S
10:05	1000k	103	7	...	0	96k	0	0	15	66	1024M	99	55	507R
10:06	1000k	103	25	...	0	96k	0	100	15	66	1024M	98	41	456S
10:07	1000k	102	19	...	0	97k	1	100	15	66	1024M	99	61	454S
10:08	1000k	102	21	...	0	97k	1	100	15	66	1024M	99	70	313R
10:09	1000k	102	20	...	0	97k	2	75	15	66	1024M	99	51	341R
10:10	1000k	103	22	...	0	97k	1	100	15	66	1024M	99	83	309R
10:11	1000k	103	31	...	0	97k	1	88	15	66	1024M	99	96	259R
10:12	1000k	103	50	...	0	97k	1	100	15	66	1024M	99	150	199R
10:13	1000k	104	251	...	0	96k	1	91	15	66	1024M	99	596	65R
10:14	1000k	106	369	...	0	95k	0	98	14	66	1024M	99	1067	32S
10:15	1000k	100	295	...	0	95k	3	98	14	66	1024M	99	819	25R
10:16	1000k	100	57	...	0	96k	0	100	14	66	1024M	99	95	40R
Enter 'GRAPHICS' command for history graphics selection menu														
Command ==>														
F1=Help F4=Top F5=Bot F7=Bkwd F8=Fwd F10=Left F11=Right F12=Return														

Table 6-4 Summary of memory usage of user IDs on the z/VM system

Note: See also Chapter 11, “Case study: slow responding Web site” on page 313, for more information about emergency scanning.

In the last window you see a summary of the memory usage of user IDs on the z/VM system. You see that our Linux running the DB2 workload has a lot of memory allocated and uses the most XSTORE pages (Table 6-5).

FCX113	CPU 2094	SER 2991E	Interval 10:17:21 - 10:18:21						Perf. Monitor	
Data >----- Number of Pages ----->										
Spaces >	<-Resident->		<--Locked-->						Stor	Nr of
Userid	Owned	>Resrld	R<2GB	R>2GB	L<2GB	L>2GB	XSTOR	DASD	Size	Users
>System<	.0	>	0	21096	19298	22	12	7778	9617	299M 24
COSTA	0	0	0	0	0	0	176	0	32M	
DATAMOVE	0	0	0	0	0	0	228	391	32M	
DIRMAINT	0	0	0	0	0	0	405	1605	32M	
DISKACNT	0	0	0	0	0	0	0	1252	32M	
DTCVSW1	0	0	1	28	1	0	20	2544	32M	
DTCVSW2	0	0	29	28	29	5	256	2517	32M	
EREP	0	0	0	0	0	0	0	1233	32M	
GCS	0	0	1	0	1	0	0	52	16M	
LNxDB2	0	0	66406	24293	5	9	96169	52155	2048M	
LNxIHS	0	0	37057	61381	42	44	21793	10563	512M	
LNxSU1	0	0	5762	8059	6	4	36900	88499	512M	
LNxSU3	0	0	1437	1220	7	6	6915	9003	64M	
LNxSU4	0	0	489	440	102	156	65	44272	256M	
LNxWAS	0	0	394602	367234	8	5	21001	3012	3072M	
MAINT	0	0	0	0	0	0	0	1401	128M	
OPERATOR	0	0	0	0	0	0	0	0	32M	
OPERSYMP	0	0	0	0	0	0	0	1267	32M	
PERFSVM	0	0	45	354	0	0	2507	3236	64M	
PVM	0	0	2	1	2	1	70	304	32M	
RSCS	0	0	2	36	1	0	0	1218	32M	
TCPIP	0	0	470	72	327	52	151	2568	32M	
VMSEVR	0	0	1	0	1	0	2	1192	32M	
VMSEVS	0	0	1	0	1	0	2	1337	64M	
Select a user for user details										
Command ==>										
F1=Help F4=Top F5=Bot F7=Bkwd F8=Fwd F10=Left F11=Right F12=Return										

Table 6-5 Linux running DB2 workload has a lot of memory allocated and uses the most XSTORE pages

Note: You can also use the **Toolkit** → **Quick Access Menu** to issue the **Toolkit** → **RXFM** → **Insert_tool** commands.

Eligibility lists

This chapter discusses the CP scheduling process and z/VM tools for problem determination. We also discuss the corrective measures that we need to take to overcome the problem with respect to the eligibility lists.

7.1 CP scheduler

When you log on to z/VM, CP creates a control block that is the anchor for a large amount of information about your virtual machine. This block is called a Virtual Machine Descriptor Block (VMDBK). It is pointers to this block that get moved around the different lists.

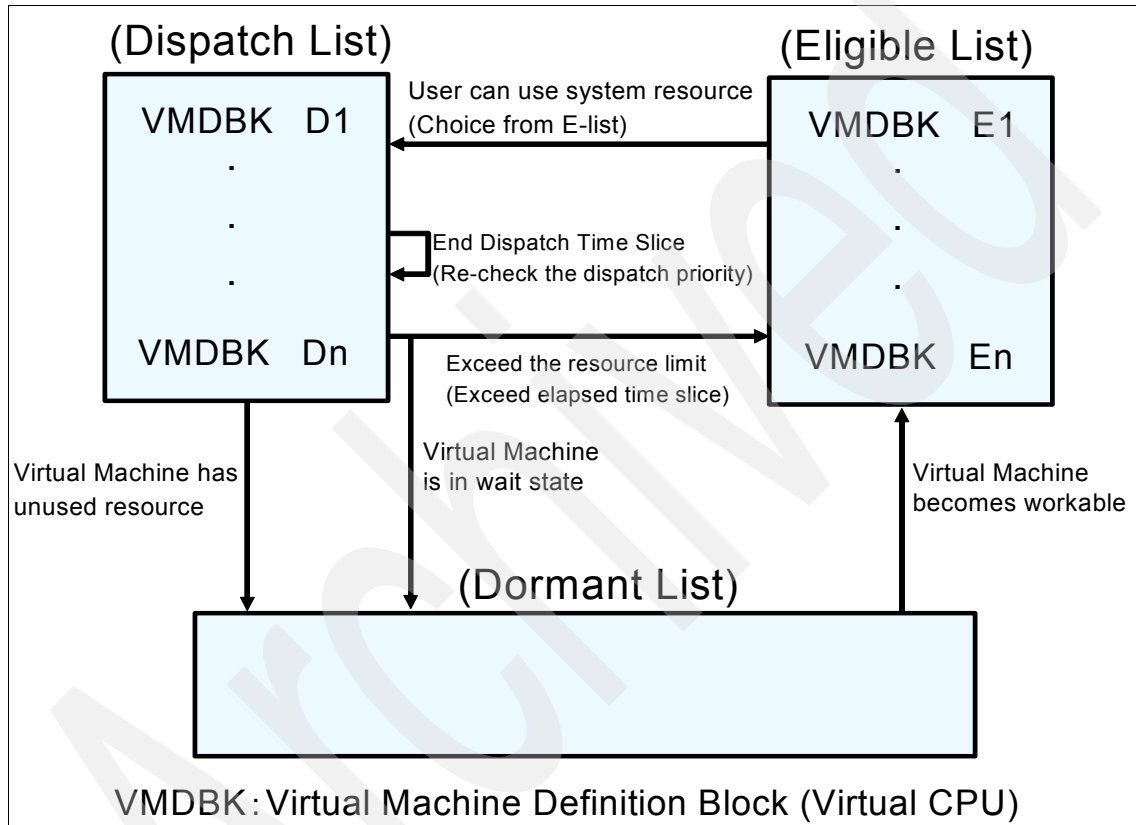


Figure 7-1 z/VM Scheduler lists

Each user is in one of the following lists:

- ▶ Dispatch list (D-list) - users ready (near-ready) to run
- ▶ Eligible list (E-list) - "Time Out" chair for users that want to be in dispatch list.
- ▶ Dormant list - users who are idle (as far as we know)

If your virtual machine is not doing anything, then CP places it in the dormant list, which implies that your virtual machine is *asleep*. If you *wake it up* (for instance, by pressing the Enter key on your console), it will get moved to the eligible list,

where it is eligible to receive system resources. Before it is given any resources, however, the virtual machine is examined by the scheduler. The scheduler looks in the VMDBK of your virtual machine to see what resources you have used in the past. Processor, storage, and paging are some of the resources examined.

If you have just logged on, there will not be much information in the VMDBK, so the scheduler thinks of you as an interactive user, also known as a class 1 user. A class 1 user is normally a CMS interactive user or a very lightly loaded guest operating system. When the scheduler determines that there is enough resource available to satisfy your needs without putting any other users in jeopardy, you will be moved to the dispatch list, where you wait in turn for your allowance of CPU time (time slice). When your turn arrives, your virtual machine is run on an available processor.

The time slice comes to an end after a certain time known as the dispatcher minor time slice. If you have more work to do, you can stay in the dispatch list and get another time slice until you have finished (in which case you will be moved to the dormant list), or you will have to pay a visit to the scheduler again so that it can reassess your resource requirements before rescheduling you.

The scheduler may decide, after you have visited the dispatch list a few times, that you are not really an interactive user but that you need more processing capacity. You will then be upgraded to class 2 user and allowed to stay in the dispatch list for longer intervals to see whether you then manage to finish your work.

This has an additional benefit because it reduces the amount of policy decisions that the scheduler has to make to manage eligible users. A class 2 user might typically be a CMS user compiling programs, or a Linux Web server.

After you have visited the dispatch list as a class 2 user several times and still not finished your work, the scheduler will upgrade you to a class 3 user and you will be allowed to stay in the dispatch list even longer to see whether you manage to finish your work. Normally, a class 3 user will be a production guest operating system.

There are drawbacks to being a higher class user. For instance, if the system resources (such as paging or storage) become constrained, then the scheduler will hold higher class users in the eligible list and let lower class users run. However, there is one type of user, known as a class 0 user (it has `OPTIONQUICKDSP` in its directory entry or has had `SET QUICKDSP` issued on its behalf) who will never be held in the eligible list. Typically, this user type is only used for important servers and special user IDs.

Table 7-1 User Classes

User class	Explanation
Class 0	This class indicates the users that were added to the dispatch list with no delay in the eligible list, regardless of the length of their current transaction.
Class 1	This class indicates the users who have just begun a transaction, and therefore are assumed to be currently processing short transactions.
Class 2	This class indicates the users who did not complete their current transactions during their first dispatch list stay and therefore are assumed to be running medium-length transactions.
Class 3	This class indicates the users who did not complete their current transactions during their second dispatch stay and therefore are assumed to be running long transactions.

If you have command privilege class E, you can issue the CP INDICATE LOAD command to view information about these classes of user. See Example 7-1. Qn indicates users in the dispatch list. En indicates users in the eligible list, where *n* is the class of the user (0, 1, 2, or 3).

Example 7-1 INDICATE LOAD command

```
ind
AVGPROC-000% 04
XSTORE-000000/SEC MIGRATE-0000/SEC
MDC READS-000001/SEC WRITES-000000/SEC HIT RATIO-084%
PAGING-0/SEC STEAL-000%
Q0-00000(00000) DORMANT-00016
Q1-00000(00000) E1-00000(00000)
Q2-00000(00000) EXPAN-001 E2-00000(00000)
Q3-00000(00000) EXPAN-001 E3-00000(00000)
PROC 0000-000% CP PROC 0001-000% CP
PROC 0002-000% CP PROC 0003-000% CP
LIMITED-00000
```

Thrashing

Thrashing is a situation caused by servers wanting more storage all at one time than exists on a box. At the point where VM does not have enough storage to meet each server's needs, VM starts paging. There is a point where VM could spend more time doing paging than performing work.

The VM scheduler has a very sophisticated mechanism to stop users from thrashing. It will control thrashing from three different perspectives: storage, paging, and processor.

The mechanism for controlling thrashing is to put users on an eligible list, meaning that these users want to consume resource, but there is not enough resource to sustain them, so they will not be dispatched. When there is enough resource or certain deadlines pass, the user will be dispatched. This can look like users are hung. They are *not* hung, they are waiting until there is sufficient resource to run efficiently.

7.2 SRM controls

Use SET SRM (system resource manager) to change system parameters. These parameters define the size of the time slice—the access to resources for different user classes as seen by the scheduler. This is a complex command and you should clearly understand its effects before using it.

7.2.1 SRM STORBUF

One of the ways to solve the thrashing problem is to set the SRM STORBUF command. One of the ways to solve the thrashing problem is to set SRM STORBUF (STORBUF). Basically, this controls the VM guest users who are not to be sent to the eligible list in spite of memory overcommitment.

The default value for SRM STORBUF is Q1=125% Q2=105% Q3=95%. Usually, users are moved to the eligible list when real storage utilization is over the setting value. When we set the values for the various queues as Q1=aaa%, Q2=bbb%, and Q3=ccc%, users are moved to the dispatch list from the eligible list with the following conditions.

- ▶ User is E1: total WSS + target E1 user's WSS \leq DPA x aaa%
- ▶ User is E2: total WSS + target E2 user's WSS \leq DPA x bbb%
- ▶ User is E3: total WSS + target E3 user's WSS \leq DPA x ccc%

WSS means working sets and DPA means dynamic paging area.

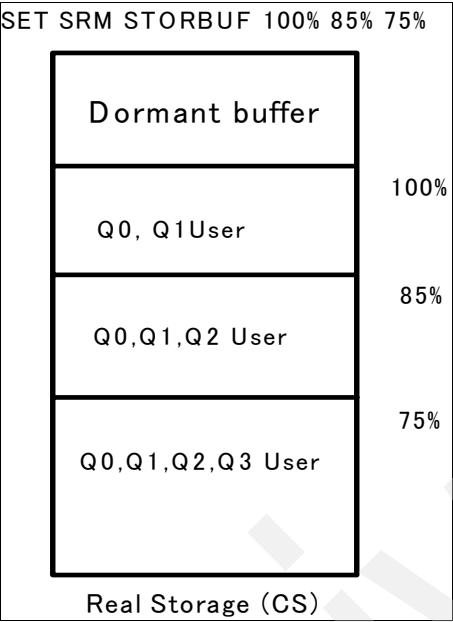


Figure 7-2 Example SET SRM STORBUF

For larger production Linux guests or servers, we recommend setting the SRM STORBUF value. See Example 7-2.

Example 7-2 Example SRM STORBUF setup for large servers

```
SET SRM STORBUF 300 250 200
```

7.2.2 SRM LDUBUF

By setting up SRM LDUBUF appropriately, we can avoid sending the users to the eligible list, and instead allow all users (E1–E3) with more paging space. The default LDUBUF encourages a thrashing situation. If you are thrashing and you raise the LDUBUF, you will in effect reduce the amount of work that you will get done.

Example 7-3 Default SRM LDUBUF values

```
LDUBUF : Q1=100% Q2=75% Q3=60%
```

In the Example 7-3 on page 238, LDUBUF figures mean that guests in Q3 are only allowed to use 60% of your paging capacity. Since Linux guests are always in Q3, we would be able to use only 60% paging capacity with our Linux guests, so we should raise the LDUBUF settings, which would mean that all guests in any queue would all be able to compete for 100% of your paging capacity. See Example 7-4.

Example 7-4 Example SRM LDUBUF setup for Linux guests

```
SET SRM LDUBUF 100 100 100
```

7.2.3 QUICKDSP

Use SET QUICKDSP to assign or unassign a user's immediate access to system resources. This option, when set for a server, allows the server to bypass all governors set by the CP scheduler. In this way, the server is always dispatched when necessary, even under a thrashing situation. Users such as TCP/IP and your more important production Linux servers should have this option.

Points to note are:

- ▶ The user ID becomes transaction class 0 and never waits in the eligible list.
- ▶ This impacts the decision to move from E-list to D-list on a user ID basis.
- ▶ Used for key server virtual machine (process synchronous requests from users) and virtual machines that demand the best performance.

Example 7-5 Syntax for SET QUICKDSP

```
SET QUICKDSP userid ON
```

The QUICKDSP setting impacts a single user ID. It should be used for any server virtual machine that is critical to user response time or holds key system resources. Overuse of this control for other users can take away the effectiveness of the scheduler.

We can also, by default, set the QUICKDSP option set for a user by including the SET QUICKDSP option in the USER DIRECTORY definition of the particular user.

7.3 Other recommendations

In general, we recommend setting QUICKDSP on for production guests and server virtual machines that perform critical system functions. However, you may

not want to set it on for all your test or non-critical guests. Allowing the VM scheduler to create an eligible list allows it to avoid some thrashing situations that could occur from over committing real storage.

Solve this with one of two approaches: use of QUICKDSP or changing the SRM STORBUF settings. The choice is dependent on where you want the responsibility of protecting the system from thrashing. The more you use QUICKDSP, the greater the responsibility on yourself. Setting appropriate STORBUF settings puts the responsibility on the VM scheduler.

The next line of defense is to set up the Linux guests conservatively as regards the virtual storage sizes and to set up the VM system well for paging. Here are some guidelines:

- ▶ Set each Linux machine's virtual storage size only as large as it needs to be to let the desired Linux applications run. This will suppress the Linux guest's tendency to use its entire address space for file cache. Make up for this with MDC if the Linux file system is hit largely by reads. Otherwise, turn MDC off, because it induces about an 11% instruction path length penalty on writes, consumes storage for the cached data, and pays off little because the read fraction is not high enough.
- ▶ Use whole volumes for VM paging, instead of fractional volumes. In other words, never mix paging I/O and non-paging I/O on the same pack.
- ▶ Implement a one-to-one relationship between paging CHPIDs and paging volumes.
- ▶ Spread the paging volumes over as many DASD control units as possible.
- ▶ If the paging control units support NVS or DASDFW, turn them on (applies to RAID devices).
- ▶ Provide at least twice as much DASD paging space (CP QUERY ALLOC PAGE) as the sum of the Linux guests' virtual storage sizes.
- ▶ Having at least one paging volume per Linux guest is a great thing. If the Linux guest is using synchronous page faults, exactly one volume per Linux guest will be enough. If the guest is using asynchronous page faults, more than one per guest might be appropriate—one per active Linux application would be more like it.
- ▶ In QDIO-intensive environments, plan that 1.25 MB per idling real QDIO adapter will be consumed out of CP below 2 GB free storage for CP control blocks (shadow queues). If the adapter is being driven very hard, this number could rise to as much as 40 MB per adapter. This tends to hit the below 2 GB storage hard. CP prefers to resolve below 2 GB contention by using XSTORE. Consider configuring at least 2 GB to 3 GB of XSTORE so as to back the below 2 GB main storage, even if main storage is otherwise large.

Example scenario

In this case, some guest operating systems are running without a problem, but other users are unable to log on. In some situations CP will be so short of a resource that it starts *thrashing*, a term that means that CP dominates the system trying to get the resources necessary to dispatch users but never gets around to actually dispatching them. The users already in the dispatch list may carry on running. Other users will show as being on the eligible list.

Example 7-6 Eligible list

```
ind load
AVGPROC-058% 03
XSTORE-000200/SEC MIGRATE-0055/SEC
MDC READS-000022/SEC WRITES-000000/SEC HIT RATIO-100%
PAGING-567/SEC STEAL-075%
Q0-00002(00000) DORMANT-00022
Q1-00005(00000) E1-00000(00000)
Q2-00009(00000) EXPAN-001 E2-00000(00000)
Q3-00012(00000) EXPAN-001 E3-00002(00000)
PROC 0000-058% CP PROC 0001-057% CP
PROC 0002-061% CP PROC 0003-058% CP
LIMITED-00000
```

This problem can be caused by insufficient storage. Use the QUERY FRAMES command to display current usage of storage. The SET SRM STORBUF command can be used to control this for different user classes if necessary, but this is more likely to be caused by either insufficient real storage or by having too many guests with very large virtual machines. Review all users to see whether any virtual machines can be made smaller.

Hardware-related problems

This chapter introduces a basic configuration of hardware and device recognition and management in Linux on System z. We describe hardware problem determination basics and introduce a case study.

8.1 Basic configuration of hardware

Figure 8-1 introduces a typical hardware configuration used by Linux on System z. If you encounter a hardware problem, this diagram may help you to understand where to look in the basic hardware configuration and connectivity to understand where your problem may lie in your own configuration.

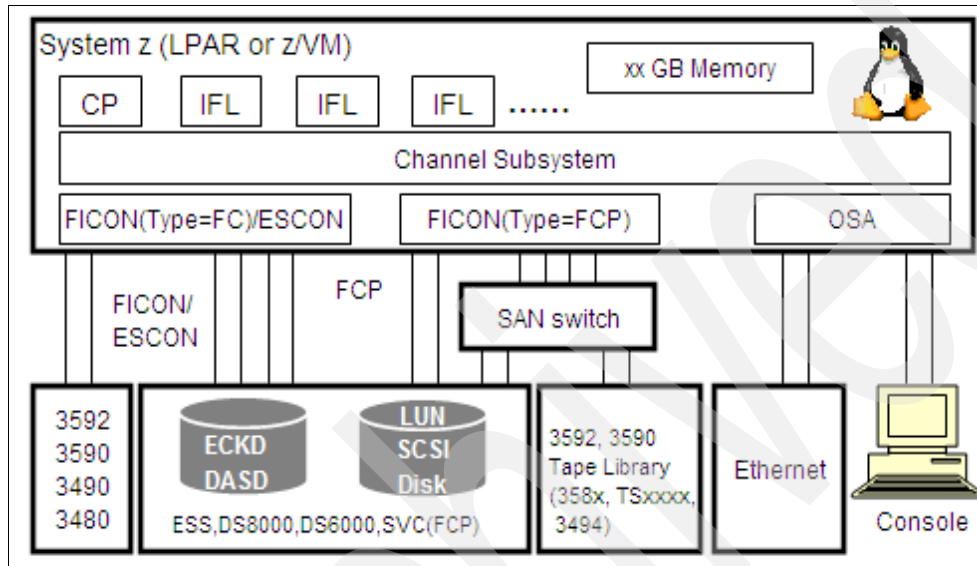


Figure 8-1 Basic hardware configuration for Linux on System z

8.2 Device recognition and management

Like other operating systems, Linux has special files that it uses to control the hardware.

This section describes devices found in Linux on System z and includes the Linux device file, sysfs, udev, device driver, common I/O, and qdio, and describes how to confirm device status.

Some of the drivers that are most often used with Linux on System z are:

- ▶ DASD (dasd_mod, dasd_eckd_mod, dasd_fba_mod)
- ▶ TAPE (tape, tape_34xxÅc.)
- ▶ FCP/SCSI (zfcp, scsi_mod, st, sg, IBMtape)
- ▶ Network (qeth)

Figure 8-2 shows of our device control.

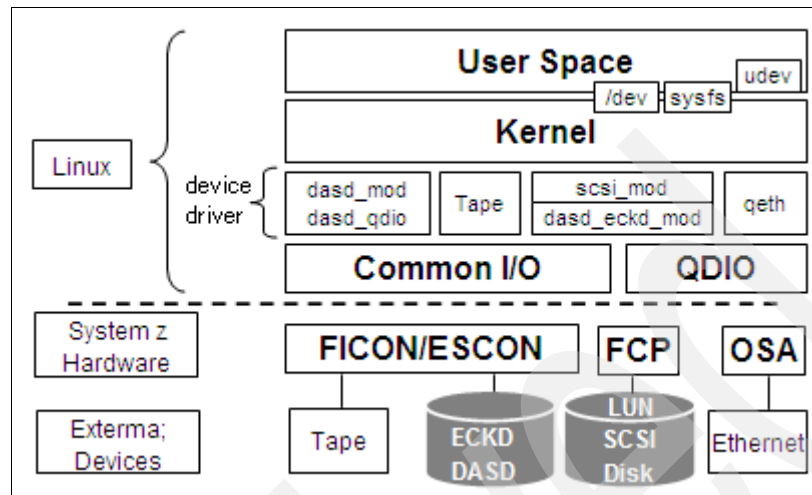


Figure 8-2 Device control

Example 8-1 on page 246 shows:

- ▶ Device drivers that are often used in Linux on System z are:
 - DASD (dasd_mod, dasd_eckd_mod, dasd_fba_mod)
 - TAPE (tape, tape_34xxÅc.)
 - FCP/SCSI (zfcp, scsi_mod, st, sg, IBMtape)
 - Network (qeth)
- ▶ The Common I/O layer is a layer that executes common I/O operations that are not dependant on devices. Actions that occur on the common I/O layer are recognition of the channel path, detection of devices, and recognition of device type and control unit type.
- ▶ The QDIO driver

Queued Direct I/O (QDIO) is a highly efficient data transfer architecture, which dramatically improves data transfer speed and efficiency for TCP/IP traffic. It provides direct communication between memory and adapter devices (Direct Memory Access, DMA). Additionally, I/O processing paths are shortened.

8.2.1 Common I/O layer

The common I/O layer is System z specific intermediate layer that exploits all hardware and kernel features. It is the layer that executes common I/O operations that are not dependent on devices. Actions that are in the common

I/O layer include recognition of the channel path, detection of devices, and recognition of device types and control unit types. Linux devices use the common I/O layer to connect to device drivers.

Common I/O processing does not depend on a device being executed. The process involved in common I/O is:

1. A starting I/O processing request is made to the channel subsystem.
2. Completing I/O - The I/O interruption from the hardware is received when I/O is completed. The subroutine of the device driver corresponding to the hardware is then called.
3. Error handling at channel and sub-channel - When the hardware event has occurred, a machine check interrupt is received. Recovery processing begins according to the event that is executed.
4. Recognizing the devices that the channel subsystem manages involves acquiring sub-channel information, acquiring the control unit, device type, and/or model number and inspection of channel paths of each device.

Overview

In this section, we provide an overview of this process from I/O request to completion and include a machine check interruption.

Channel command word

Channel command words (CCWs) are I/O requests to devices and are hardware understandable instructions. The Information included in the CCWs is:

- ▶ Hardware instruction such as read or write
- ▶ Data or buffer addresses on memory
- ▶ Number of bytes for input/output data

Instructions can be different for each device.

Channel report word messages

Channel report word (CRW) messages provide a status of the I/O operation, the device, and the channel paths (for example, link up/link down message or messages about attaching devices). An example of a message is shown in Example 8-1, where a FICON cable is pulled out once and reconnected.

Example 8-1 /var/log/messages

```
Jul 1 15:03:35 xxxxxxxx kernel: crw_info : CRW reports slct=0, oflw=0,
chn=0, rsc=B, anc=0, erc=0, rsid=0
Jul 1 15:03:36 xxxxxxxx kernel: dasd_erp: 0.0.c700: default ERP called
(255 retries left)
```

Figure 8-3 demonstrates the I/O request, a machine check interruption, and the I/O completion.

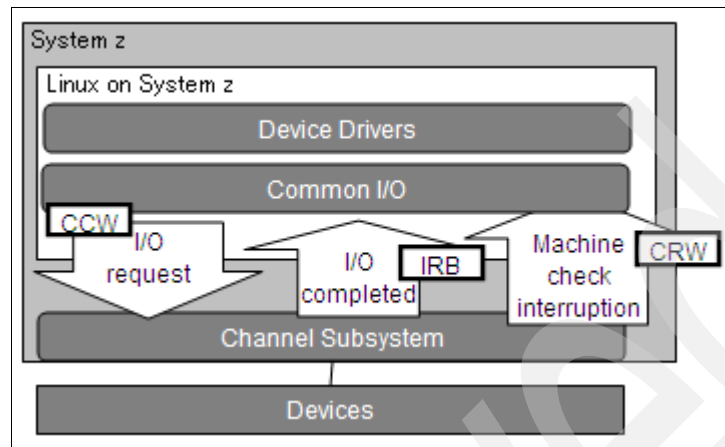


Figure 8-3 Overview from I/O request to completion

During the I/O request, the CCW is created and it executes start sub channel (SSCH) instruction.

When the I/O is completed, notification is sent to the operating system by an I/O interruption. The status is stored to the interruption response block (IRB). The IRB stores device and channel status.

During a machine check interruption, the CRW provides notification of hardware events such as channel links up or down.

8.2.2 Kernel layer

The Linux device drivers work through special kernel code that accesses the hardware directly. This kernel layer recognizes and manages the hardware with special device files such as sysfs and udev. Figure 8-4 shows where these devices sit in the overall architecture.

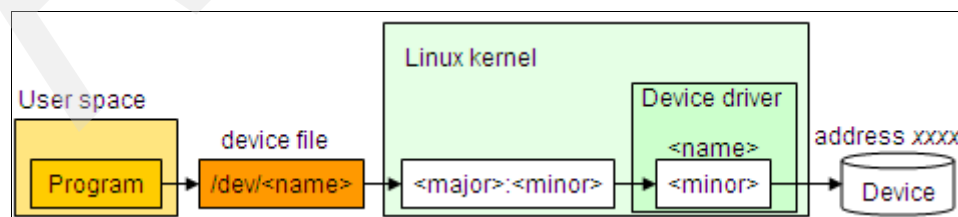


Figure 8-4 Device drivers and the Linux kernel

Linux treats a device as a file. As seen in Figure 8-4 on page 247, in the user space is a program that controls devices using device files. The Linux kernel manages the devices using major and minor numbers. The major number is a number that indicates the device driver and the minor number is a number that indicates each device.

Some device files are created automatically when Linux is installed. From kernel Version 2.6 forward, Linux creates device files using a hardware detector called udev, which also manages file permissions. udev works in the user space using hotplug events that the kernel sends whenever a device is added or removed from the kernel. udev creates device files both automatically and manually.

When a new device arrives, the hardware or z/VM creates a machine check. The Linux machine check handler handles the machine check while the Linux common I/O layer queries the channel subsystem. The new device is registered in the Linux device infrastructure where a new sysfs entry appears and a hotplug event is created. udev examines the configuration files and creates the device node appropriately.

By creating and dynamically deleting device files, device files of only devices that are connected to the system are created under the /dev directory, and it is easy to confirm what devices are connected now and whether the devices are connected correctly. However, in SLES, minimum device files must exist in the system, and these files are not deleted dynamically.

The steps involved in device recognition by udev are shown in Figure 8-5.

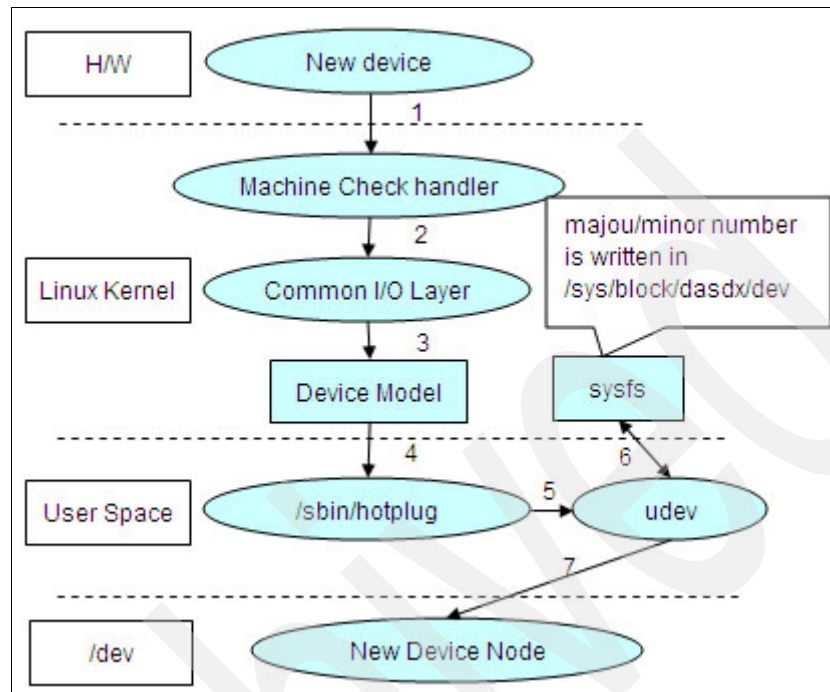


Figure 8-5 Sequence of attaching new devices with hotplug

The steps are:

1. A new device is added and a machine check occurs.
2. The machine check handler informs the common I/O layer of the new device.
3. Information about the added device is acquired in the common I/O. A new entry is created in sysfs.
4. Hotplug is called.
5. udev is called.
6. udev references sysfs.
7. The device file is created.

Sysfs offers an interface to the user space for accessing the data structure in the kernel space. Sysfs can dynamically add and delete via both the hotplug agent and by user operations. It also allows for option setting (log level, for example) of the device and driver and confirmation of device status (such as on or offline).

The Sysfs contains virtual configuration and control files, for example:

- ▶ Example of configuring an OSA device

```
# echo "0.0.0769,0.0.076a,0.0.076b" >
/sys/bus/ccwgroup/drivers/qeth/group
# echo 1 > /sys/bus/ccwgroup/drivers/qeth/0.0.0769/online
```
- ▶ Example of the changing log level

```
# echo 2 > /sys/bus/ccw/drivers/zfcp/loglevel_qdio
```
- ▶ Example of making channel path offline/online

```
# echo offline > /sys/devices/css0/chp0.xx/status
# echo online > /sys/devices/css0/chp0.xx/status
```
- ▶ Example of changing option setting

```
# echo 1 > /sys/devices/qeth/0.0.e18c/layer2
```

8.2.3 Device drivers

The following are the device drivers, their corresponding kernel modules (if applicable), and descriptions and characteristics that are common in a Linux on System z configuration:

- ▶ DASD device driver
 - Kernel modules: `dasd_mod`, `dasd_eckd_mod`, `dasd_fba_mod`.
 - A device driver that is used to access channel-attached DASD devices.
 - Channel subsystem processes path control and error recovery.
Linux does not consider the existence of path.
 - Linux controls it by using device file `/dev/dasdxÅh`.
 - The first partition is `/dev/dasdx1`, the second partition is `/dev/dasdx2`.
 - The maximum number of partitions is 3.
- ▶ zfcp device driver
 - A device driver that is used to access Fibre Channel attached SCSI devices.
 - A channel subsystem is not used. The buffer I/O of QDIO is used instead. Linux has to process path control and error recovery by itself because of this.
 - Information about the Host Bus Adapter (HBA) and the logical unit number (LUN) on the storage server side is needed in order to recognize a LUN.
 - Linux controls recognize the LUN by using the device file, `/dev`.

- Network device driver - Network devices are controlled by network device drivers.
 - The device file and major/minor numbers are created in the same way as disk and tape.
 - The interface is created and related to a device.
- The default name of an interface depends on the type of interface, for example, Ethernet (eth0, eth1, HiperSockets (hsi0, hsi). The name of the interface can be changed <qeth>.
- Device drivers are used to access the OSA-Express of QDIO mode and HiperSockets.
- Channel subsystems are not used. Buffer I/O of QDIO is used instead.
- The qeth device driver needs 3 subchannels to recognize an OSA device:
 - An even number of control read devices.
 - The control write devices must be the device bus-ID of the read subchannel plus one.
 - Data devices can be any free device bus-ID on the same CHPID.

8.2.4 Confirming status of channel path and devices

This section describes the command and files for status confirmation of devices. It is possible to confirm the devices' status provided by sysfs. It is important to confirm the status, even under normal circumstances.

Table 8-1 Confirming status of channel path and devices

	Commands/files	Description and contents
General	lscss	The device status that common I/O manages
	/etc/sysconfig/hardware/*	Device information
DASD	lsdasd	Composition information about DASD
	/sys/bus/ccw/drivers/dasd-eckd/0.0.<addr>	Composition information about DASD
	/proc/dasd	Composition information and statistical information about DASD
FCP	lsscsi	Information on SCSI devices
	/sys/bus/ccw/drivers/zfcp/0.0.<addr>	Composition information about SCSI devices
	/proc/scsi/IBMtape, /proc/scsi/IBMchanger	FCP tape device information

	Commands/files	Description and contents
Network	lsqeth	Composition information on the qeth network devices
	/sys/bus/ccwgroup/drivers/qeth/0.0.<addr>	qeth network devices information
	/proc/qeth	qeth network devices information

It is possible to confirm the status of the channel path and devices by using the `lcss` command. See Example 8-2.

Example 8-2 Confirming channel path status and device status

Device	Subchan.	DevType	CU	Type	Use	PIM	PAM	POM	CHPIDs	
0.0.0009	0.0.0000	0000/00	3215/00	yes	80	80	FF	00000000	00000000	
0.0.000C	0.0.0001	0000/00	2540/00		80	80	FF	00000000	00000000	
0.0.000D	0.0.0002	0000/00	2540/00		80	80	FF	00000000	00000000	
0.0.000E	0.0.0003	0000/00	1403/00		80	80	FF	00000000	00000000	
0.0.0190	0.0.0004	3390/0A	3990/E9		F0	F0	FF	C0C2C1C3	00000000	
0.0.019E	0.0.0005	3390/0A	3990/E9		F0	F0	FF	C0C2C1C3	00000000	
0.0.019D	0.0.0006	3390/0A	3990/E9		F0	F0	FF	C0C2C1C3	00000000	
0.0.0191	0.0.0007	3390/0A	3990/E9		F0	F0	FF	C0C2C1C3	00000000	
0.0.05B0	0.0.0008	1732/01	1731/01	yes	80	80	FF	A9000000	00000000	
0.0.05B1	0.0.0009	1732/01	1731/01	yes	80	80	FF	A9000000	00000000	
0.0.05B2	0.0.000A	1732/01	1731/01	yes	80	80	FF	A9000000	00000000	
0.0.0240	0.0.000B	1732/01	1731/01	yes	80	80	FF	BE000000	00000000	
0.0.0241	0.0.000C	1732/01	1731/01	yes	80	80	FF	BE000000	00000000	
0.0.0242	0.0.000D	1732/01	1731/01	yes	80	80	FF	BE000000	00000000	
0.0.C128	0.0.000E	3390/0A	3990/E9	yes	F0	F0	FF	C0C2C1C3	00000000	
0.0.C129	0.0.000F	3390/0A	3990/E9	yes	F0	F0	FF	C0C2C1C3	00000000	
0.0.38CD	0.0.0014	1732/03	1731/03		80	80	FF	B6000000	00000000	

Channel path status

You would confirm the physical path status by using the `lscss` command and looking at the columns labeled PIM, PAM, and POM. These are a part of the path status information that are included in the subchannel information block. See Figure 8-6.

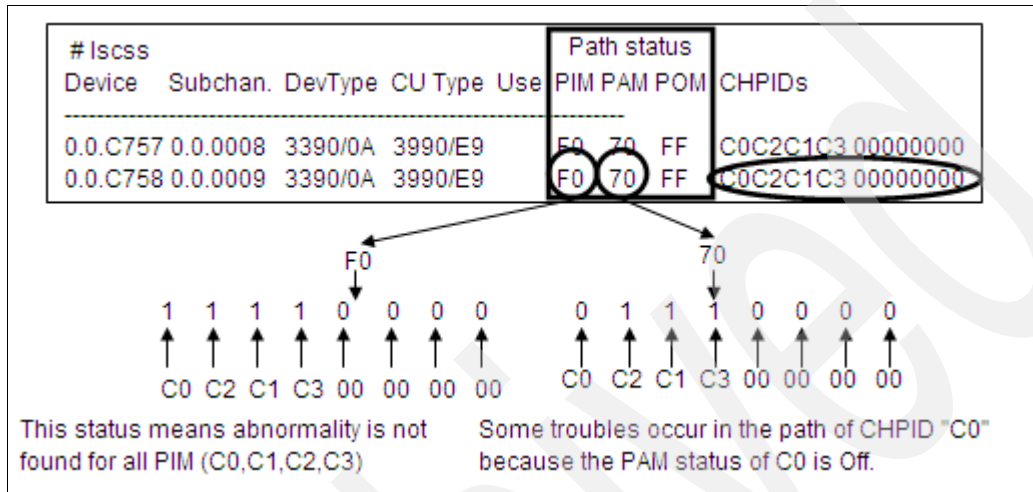


Figure 8-6 Confirming channel path status

Confirming device status

To confirm the status of DASD, use the `lsdasd` command found in `/proc/dasd/devices`.

Output from this command will be similar to that found in Example 8-3.

Example 8-3 Confirming DASD status

```
0.0.c128(ECKD) at (94: 0) is dasda : active at blocksize 4096, 601020 blocks, 2347 MB
0.0.c129(ECKD) at (94: 4) is dasdb : active at blocksize 4096, 601020 blocks, 2347 MB
0.0.c12a(ECKD) at (94: 8) is dasdc : active at blocksize 4096, 601020 blocks, 2347 MB
```

To confirm the FCP device status, use the **lsscsi** command, as shown in Example 8-4.

Example 8-4 Confirming FCP device status

```
# lsscsi -v
sysfsroot: /sys
[0:0:1:0]    disk      IBM      2105800      .112  /dev/sda
dir: /sys/bus/scsi/devices/0:0:1:0
[/sys/devices/css0/0.0.0007/0.0.f441/host0/0:0:1:0]
```

To confirm OSA status, use the **lsqeth** command (this command is only for SLES), as shown in Example 8-5.

Example 8-5 Output of the lsqeth command - confirming OSA devices

Device name	: eth0

card_type	: GuestLAN QDIO
cdev0	: 0.0.05a0
cdev1	: 0.0.05a1
cdev2	: 0.0.05a2
chpid	: A9
online	: 1
portno	: 0
route4	: no
route6	: no
checksumming	: sw checksumming
state	: UP (LAN ONLINE)
priority_queueing	: always queue 2
fake_ll	: 0
fake_broadcast	: 0
buffer_count	: 16
add_hhlen	: 0
layer2	: 0
large_send	: no

8.3 Hardware problem determination

This section provides:

- ▶ An explanation of check points that will help you determine whether a problem is hardware or software.
- ▶ A procedure to determine each step of the problem of hardware recognition and to determine the problem.

8.3.1 Initial determination of hardware or software problems

If a problem happens, you will want to confirm the problem first. Gather `dbginfo.sh` and `s390dbf` first. Determine the following:

- ▶ Are the devices be recognized?
- ▶ Is the performance up to expectations?
- ▶ Can the devices be used?

Even though there may be no problem in device recognition and performance, some errors may still be reported, look in the `/etc` directory.

After confirming the problem, look at the components that can be easily observed (such as, is it storage subsystem trouble? network trouble?). Determine whether it is a problem related to I/O. I/O troubles should be investigated last.

Messages that may indicate a hardware problem are:

- ▶ Remote Support Facility (RSF) notification
- ▶ The icon status of the HMC
- ▶ Check stop of LPAR

Distinct hardware or software problem

To determine whether it is clearly a problem with hardware or software, error messages and problem scope (single system or multi system), leave good hints. Hardware error messages are shown in Linux logs, on the HMC, on the z/VM console, or in the IBM z/VM Environmental Record Editing, and Printing (EREP) report. A flowchart of the problem determination process can be seen in Figure 8-7.

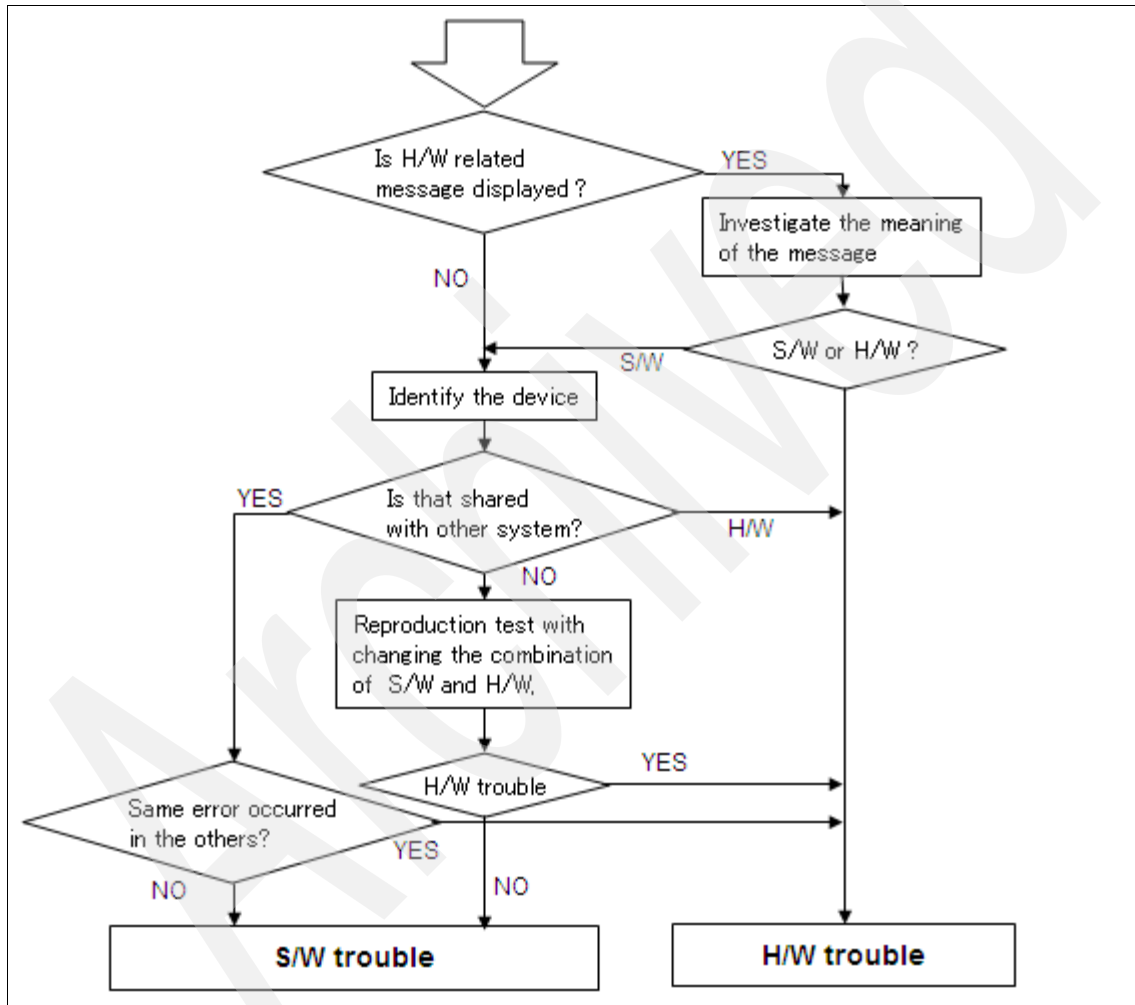


Figure 8-7 Hardware and software problem determination flowchart

Note: /var/log/messages on Linux does not have the messages that are shown during Linux hang, down, or reboot.

8.3.2 Investigating hardware-related messages

Hardware error messages in z/VM basically have an error code and its meaning. Check the messages and code manual of each component for your version of z/VM.

We have no manual about hardware messages in Linux. Searching the meaning of messages from past cases is a good approach of investigation. You can search in the z/VM Technical Q&A library and Bugzilla. Linux source code would have its meaning.

Here we provide some sample hardware error messages.

Example of hardware error messages for DASD

Sense data is an error returned from a SCSI device. Sense data is available in the sense buffer. For more information about sense data, see:

<http://www.linux.org/docs/ldp/howto/archived/SCSI-Programming-HOWTO/SCSI-Programming-HOWTO-10.html>

The sense data description is displayed directly in the message. Example 8-6 shows a sample of this sense data.

Example 8-6 /var/log/messages

```
Dec 18 19:19:04 xxxxxxxx kernel: dasd_erp(3990): 0.0.2008: Environmental data present
Dec 18 19:19:04 xxxxxxxx kernel: dasd_erp(3990): 0.0.2008: FORMAT F - Operation
Terminated
Dec 18 19:19:04 xxxxxxxx kernel: dasd_erp(3990): 0.0.2008: dasd_3990_erp_action_4:
first time retry
Dec 18 19:19:04 xxxxxxxx kernel: dasd_erp(3990): 0.0.2008: waiting for state change
pending interrupt, 255 retries left
Dec 18 19:19:04 xxxxxxxx kernel: dasd_erp(3990): 0.0.2008: blocking request queue for
30s
```

Another example of a hardware error message concerning DASD would be an equipment check error.

- ▶ Equipment check (EQC) usually indicates a hardware error.
- ▶ The message in Example 8-7 indicates an EQC of a reset event. A Reset Event is returned from control units on the timing of the first access after a channel path reset. The channel path is reset when the reconnection has occurred after the physical connection of ESCON® or FICON is disconnected.
- ▶ You will not see that the connection is broken when the message is displayed.

Example 8-7 /var/log/messages

```
Feb  6 01:07:19 xxxxxxxx kernel: dasd_erp(3990): 0.0.7404: Equipment Check -
environmental data present
Feb  6 01:07:19 xxxxxxxx kernel: dasd_erp(3990): 0.0.7404: FORMAT 0 - Reserved
Feb  6 01:07:19 xxxxxxxx kernel: dasd_erp(3990): 0.0.7404: dasd_3990_erp_action_4:
first time retry
Feb  6 01:07:21 xxxxxxxx kernel: dasd_erp(3990): 0.0.7403: Equipment check or
processing error
```

Example of error messages for FCP devices

In Example 8-8, we see an error of the tape drive with an FCP connection.

The following messages are output during backup to tape. If this happens, check the address of the adapter from the error messages at first. The following list refers to Example 8-8:

- ▶ Discover Address (ADISC) Request failed.
- ▶ The port is not connected with SAN network.
- ▶ Recovery failed.

Example 8-8 /var/log/messages

```
Apr  3 07:07:21 xxxxxxxx kernel: zfcpl: ELS request rejected/timed out, force physical
port reopen (adapter 0.0.1902, port d_id=0x00390001) (1)
Apr  3 07:07:52 xxxxxxxx kernel: zfcpl: The adapter 0.0.1902 reported the following
characteristics:
Apr  3 07:07:52 xxxxxxxx kernel: WWNN 0x5005076400cd191f, WWPN 0x50050764016077b7,
S_ID 0x00390010,
Apr  3 07:07:52 xxxxxxxx kernel: adapter version 0x3, LIC version 0x606, FC link
speed 2 Gb/s
Apr  3 07:07:52 xxxxxxxx kernel: zfcpl: Switched fabric fibrechannel network detected
at adapter 0.0.1902.
```

```
Apr  3 07:07:52 xxxxxxxx kernel: zfcpx: warning: failed gid_pn nameserver request for  
wwpn 0x500507630f400d04 for adapter 0.0.1902 (2)  
Apr  3 07:07:52 xxxxxxxx kernel: zfcpx: port erp failed (adapter 0.0.1902,  
wwpn=0x500507630f400d04) (3)
```

ELS in the example refers to extended link service. In this case, ELS requests the address information of the target port.

gid_pn refers to Get Port Identifier - Port Name. Send the port name to a name server and get the port ID.

Examples of messages that look like hardware errors

In this example, we are trying to recognize DASD that is not formatted.

The following messages (shown in Example 8-9) are displayed, but this does not mean that it is a hardware error. The error messages would not be displayed if the DASD were formatted.

Example 8-9 /var/log/messages

```
Mar 17 16:01:05 XXXXX kernel: unable to read partition table  
dasdaa:<3>dasd_erp(3990): /dev/dasdaa ( 94:104),29d0@2b: EXAMINE 24: No Record  
Found detected - fatal error  
dasd(eckd): Sense data:  
dasd(eckd):device 29D0 on irq 43: I/O status report:  
dasd(eckd):in req: 01a04600 CS: 0x40 DS: 0x0E  
dasd(eckd):Failing CCW: 01a046d0  
dasd(eckd):Sense(hex) 0- 7: 00 08 00 00 d0 00 00 00  
dasd(eckd):Sense(hex) 8-15: 00 00 00 00 00 00 00 05  
dasd(eckd):Sense(hex) 16-23: 27 00 72 80 20 0a 0f 00  
dasd(eckd):Sense(hex) 24-31: 00 00 40 e2 00 00 00 00  
dasd(eckd):24 Byte: 0 MSG 0, no MSGb to SYSOP
```

Example 8-10 is another example of a message that looks like a hardware error. It is displayed when the ext3 file system is used. The transaction barrier function is not supported by the storage systems.

The transaction barrier function is a supplementary mechanism for keeping data integrity.

To avoid this, simply add the parameter “barrier=off” to the file ziapl.conf.

Example 8-10 /var/log/messages

```
JBD: barrier-based sync failed on dasdxx - disabling barriers
```

8.3.3 Approach for hardware problems

The approach that you should take to solving hardware problems would be to first contact your hardware support personnel. You will need the hardware support person to further investigate the hardware problem.

Some basic information that they will need is:

- ▶ Machine type
- ▶ Serial number
- ▶ Telephone number of the machine room

Information that is necessary to provide to your support person is:

- ▶ A description of the problem
- ▶ What you were doing when the problem occurred
- ▶ When it occurred
- ▶ Whether you changed any of the settings just before the problem occurred
- ▶ The information to specify the hardware
- ▶ Information about the hardware (for example, address)
- ▶ Error messages on Linux
- ▶ The information that you can get from HMC

Specify the device

If we can specify the device in error clearly, the following two cases are possible.:

- ▶ One device is in error.
- ▶ The device status is not normal.

If we cannot specify the device in error, then:

- ▶ The performance trouble could be disk I/O, which may be on different storage systems.
- ▶ There may be network connection problems in an environment that has multiple NICs.

After specifying the device in error, check the status and distinguish the error (hardware or software).

Check the status of DASD recognition

Next you will want to check the status of DASD recognition by checking the following:

- ▶ The Common I/O
Check **lscss** output.
 - Check whether the address of the target device is displayed.
 - Check whether DevType and CU type are collect.
- ▶ The device driver
Check the `/sys/bus/ccw/drivers/dasd-eckd` directory.
Check whether the symbolic link of the target device is in `/sys/bus/ccw/drivers/dasd-eckd`.
- ▶ Online status
 - Check **lsdasd**.
 - Check **lscss**.
 - Check `/sys/bus/ccw/drivers/dasd-eckd/0.0.<address>/online`.

Check the status of FCP device recognition

You will want to check the status of your FCP device recognition:

- ▶ FCP port on System z
Check the **lscss**.
Use="yes" means that the FCP port is online on System z.
`/sys/bus/ccw/drivers/zfcp/0.0.xxxx/online` means the same.
- ▶ FCP port on storage system
Check the `/sys/bus/ccw/drivers/zfcp/0.0.xxxx/` directory.
If is a directory that is named with WWPN, FCP port is online on the storage system.
- ▶ LUN recognition
Check the `/sys/bus/ccw/drivers/zfcp/0.0.xxxx/yyy...yyy/` directory (yyy...yyy is WWPN).
There is a directory that is named with the LUN number, meaning that the LUN is recognized.
- ▶ SCSI disk recognition
 - Check **lsscsi**.
 - Check `/proc/scsi/scsi`.

Check the status of OSA recognition

You will want to check the status of your OSA recognition:

- ▶ Common I/O
Check **lscss**.
- ▶ Device driver
 - Check the symbolic link about devices on `/sys/bus/ccwgroup/drivers/qeth/*`.
 - Check the entry on `/proc/qeth`.
- ▶ Online status
 - Check **lscss**.
 - Check the status on `/proc/qeth`.

Investigation of shared devices

In many Linux on System z environments, system resources (CPU, memory, channel, network) are shared. When hardware trouble happens, the same problem will occur on all systems that share the hardware.

If it is difficult to distinguish whether the problem is related to hardware or software, additional tests with changing software versions or combinations of hardware and software may be necessary.

- ▶ Change the software version.
If the trouble does not occur when the software or hardware version is changed, then the software is the problem.
- ▶ Assign the same hardware to the other system.
 - If the trouble occurs on the other system, then hardware is the problem.
 - If the trouble does not occur on the other system, then software is the problem.
- ▶ Assign different hardware to the same system.
 - If the problem occur on the other hardware, then software is the problem.
 - If the problem does not occur on the other hardware, then hardware is the problem.

8.3.4 After determining whether problem is software or hardware

If it seems that software is the problem:

- ▶ Determine whether it is in the device driver layer.
- ▶ Determine whether it is on each component.

If it seems that hardware is the problem:

- ▶ Contact hardware support personnel.
- ▶ Prepare the procedure for making a hardware exchange.

If you cannot distinguish whether it is a software or a hardware problem, reproduce the problem and gather information:

- ▶ Confirm the problem reproduction procedure.
- ▶ Set the proper debug level in s390dbf and other log and trace.
- ▶ Run the problem reproduction test.
- ▶ Get information from dbginfo.sh or other tools.

8.4 Case study

This section provides case studies of hardware problems.

8.4.1 Memory card

The characteristics of memory card problems are:

- ▶ Memory error messages are displayed on the HMC.
- ▶ Notice from remote support facility (RSF).
- ▶ Check stop or system hung occurs in some LPARs that share the same memory card.
- ▶ Environment: RHEL4 update 4 on System z9® LPAR.
- ▶ Information gathered: System messages from HMC OS messages.
- ▶ Analysis step: Contact a hardware support person.
- ▶ Result and solution: Hardware problem. Exchange the memory card.
- ▶ Tips: If RSF is reported, we can determine that the trouble is a hardware problem.

8.4.2 OSA

The characteristics of OSA problems would be:

- ▶ State: Network interface goes down with qeth error messages (Example 8-11).

Example 8-11 /var/log/messages

```
qeth: check on device 0.0.3600, dstat=x0, cstat=x4 <4>
qeth: irb: 00 c2 40 17 03 77 50 38 00 04 00 00 00 00 00 00
qeth: irb: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
qeth: irb: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
qeth: irb: 00 00 00 00 00 00 00 00 00 00 02 00 15 00 5d
qdio : received check condition on activate queues on device 0.0.3602
(cs=x4, ds=x0).
qeth: Recovery of device 0.0.3600 started ...
bonding: bond1: link status definitely down for interface eth2,
disabling it
bonding: bond1: making interface eth3 the new active one.
```

- ▶ Environment: SLES9 SP3 on System z9 (not related on LPAR or on z/VM).
- ▶ Information gathered: dbginfo.sh (especially /var/log/messages).
- ▶ Analysis step
 - Subchannel status on /var/log/messages shows subchannel status (channel control check).
“cstat” on “qeth: check on device...” shows subchannel status.
 - Channel control check is reported when hardware problem happens.
 - Contact your hardware support person.
- ▶ Result and solution: hardware problem. Changing the OSA card will solve the problem.
- ▶ Tips: Device status or subchannel code often shows hints that help distinguish between hardware and software problems.

8.4.3 DASD errors

The characteristics of DASD errors are:

- ▶ State: DASD error messages in /var/log/messages (see Example 8-12).

Example 8-12 /var/log/messages

```
Apr 19 15:14:09 xx kernel: dasd: 0.0.0104: Interrupt fastpath failed!
Apr 19 15:14:09 xx kernel: dasd: 0.0.0106: Interrupt fastpath failed!
Apr 19 15:14:09 xx kernel: dasd_erp: 0.0.0104: default ERP called (254 retries left)
Apr 19 15:14:09 xx kernel: dasd_erp: 0.0.0106: default ERP called (254 retries left)
Apr 19 15:14:09 xx kernel: dasd_erp: 0.0.0100: default ERP called (255 retries left)
Apr 19 15:14:09 xx kernel: dasd_erp: 0.0.0103: default ERP called (255 retries left)
Apr 19 15:14:10 xx kernel: dasd: 0.0.010c: Interrupt fastpath failed!
```

- ▶ Environment: SLES9 SP3 on System z (not related on LPAR or on z/VM).
- ▶ Information gathered: dbginfo.sh (especially /var/log/messages).
- ▶ Analysis step
 - Search the error messages. Messages report an error when I/O requests are performed.
 - Error messages are displayed in some device address on the same path. Confirm that this is a hardware error or a hardware change with hardware support person.
- ▶ Result and solution: hardware temporary error. ESS detected the temporary error of the FC adapter at the same time. ESS does a warm start (about 2 seconds). If there is no longer a problem, ESS has recovered normally.
- ▶ Tips: It appears to be a path error if the error occurs in some address on the same path.

8.4.4 Network interface error upon reboot

The characteristics of a network interface error upon reboot are:

- State
 - No interface found message of network interface is displayed in boot.msg after reboot (Example 8-13).
 - The bonding interface is not configured.
 - The same phenomenon is occurring in some LPARs that share the OSA port.

Example 8-13 /var/log/boot.msg

[1A..done	qeth-bus-ccw-0.0.3700	No interface found
[1A..failed	qeth-bus-ccw-0.0.3700	No interface found
[1A..failed	bond0	
	bond0	enslaving interfaces: eth0 eth1
	bond0	IP address: x.xx.xx.x/24 as bonding master
[1A..done	bond1	
	bond1	Could not get an interface for slave device
	'qeth-bus-ccw-0.0.3700'	
	bond1	enslaving interfaces: eth2
	bond1	IP address: x.xx.xx.x/24 as bonding master

- Environment: SLES9 SP3 on System z (not related on LPAR or on z/VM).
- Information gathered: dbginfo.sh.
- Analysis steps: The same phenomenon is occurring in some LPARs that share OSA ports, so it seems to be a hardware problem.
 - c. Your hardware support engineer confirms that the hardware status is normal.
 - d. You learned of the device failure from the lscss.
“DevType”, “CU Type” is received as response to senselD from the hardware.
The cause is one of the following:
 - SenselD is not sending to the device.
 - SenselD is sent, but OSA does not reply.
 - e. Check the process of device initialization from the source code and trace (by support center). SenselD is sent when LACP turns ON.

- f. Try to get the debug trace in the environment in which it is occurring. Set the debug level:

```
# echo 6 > /proc/s390dbf/cio_msg/level
# echo 6 > /proc/s390dbf/cio_trace/level
```

- g. Set LCHP OFF/ON

```
# echo "offline" > /sys/devices/css0/chp0.63/status
# echo "online" > /sys/devices/css0/chp0.63/status
```

- h. Get dbginfo.

If after LCHP is turned off and the status still does not change to online:

- a. Examine hardware again.
- b. Forcibly obtain the detailed information from the OSA. (Use `forcelog`, which are error logs obtained by the operating system engineer.) Check that the stop happens.

- Result and solution: hardware problem.

Exchanging the OSA card will solve the problem.

- Tips: The same phenomenon is occurring in some LPARs that share an OSA port.

It seems more likely to be a hardware problem.

When it is difficult to determine whether it is hardware trouble or software trouble, reproduce the problem and get the `s390dbf`.

8.4.5 DASD recognition problem in boot process

The characteristics of a DASD recognition problem in the boot process are:

- State: The system does not recognize the DASD in reboot (see Example 8-14).

Example 8-14 /var/log/boot.msg

```
Couldn't find device with uuid 'cuPnQ1-zYo0-3bo5-gxrc-b5VL-Nomg-prT5qf'.
Couldn't find all physical volumes for volume group vgHEPRRC01arcdest0401.
/dev/dasdc1: lseek 0 failed: invalid argument
/dev/dasdc1: lseek 0 failed: invalid argument
```

- Environment: SLES9 on System z.
- Information gathered:
- /var/log/boot.msg
 - `dbginfo.sh` (includes /var/log/messages)
 - `sigano`

- Stand-alone dump
- Analysis step:
 - DASD is recognized as non-label according to /var/boot.msg (Example 8-15).
 - Size of DASD partition's size is abnormal on /proc/partitions in dbginfo (Example 8-16).

Example 8-15 /var/log/boot.msg

```
dasd(eckd): 0.0.747f: 3390/0C(CU:3990/01) Cyl:10017 Head:15 Sec:224
dasd(eckd): 0.0.747f: (4kB blks): 7212240kB at 48kB/trk compatible disk
dasda1:VOL1/ 0X747F: dasda11
dasd(eckd): 0.0.77a9: 3390/0C(CU:3990/01) Cyl:10017 Head:15 Sec:224
dasd(eckd): 0.0.77a9: (4kB blks): 7212240kB at 48kB/trk compatible disk
dasdca:(nonl)/      : dasdca1
```

Example 8-16 /proc/partitions

94	149	7212144	dasda11
94	312	7212240	dasdca
94	313	9223372036854775796	dasdca1
94	244	7212240	dasdbj

- Search the flow of I/O processes from s390dbf in dump.
 - Search by support center and developer.
 - The function for DASD recognition does not get the volume label information and partition information.
- Result and solution: software problem.
 - It is the same problem as reported in SLES10.
 - Use the release patch for the current kernel of SLES9.
- Tips: Determine that this problem is the same as that of SLES10 according to an examination of /var/log/boot.msg, dbginfo, and dump information.

Installation and setup problems

This chapter is a guide to help you to identify most problems that you could encounter during the installation and setup process. This chapter will help you to make the problem determination much easier.

This chapter covers the following:

- ▶ Understanding the installation process
- ▶ Gathering information regarding installation
- ▶ Scenarios

9.1 Understanding the installation process

Before starting a Linux installation we need to check for system requirements (for example, memory, network, disk space). To understand the details of these requirements, see the Web site of your preferred Linux distributor, such as:

<http://www.novell.com>

<http://www.redhat.com>

A Linux installation on System z can be performed in two ways:

- ▶ Under z/VM - When Linux is installed on z/VM, the Linux distribution is considered a guest system.
- ▶ On a System z LPAR - When Linux is installed using LPAR, permit it to use a part of physical memory on the system.

It is possible to install Linux on System z either from graphical interface or from a text interface.

If we decide to use a graphical user interface, you can use your mouse to navigate the screens, enter text fields, or click buttons.

To run the graphical installation, we can use either of two options below from your Linux or Windows® workstation machine. You must also have another server running.

- ▶ X11 forwarding

From your Linux machine, open a command prompt and type the following:

```
$ ssh -X servername.company
```

Where:

- -X is the option that enables X11 forwarding. This can also be specified on a per-host basis in a configuration file.
- servername.company is a server machine where Linux will be installed (for example, lnx.itso.ibm.com).

- ▶ VNC Client

From your Linux or windows machine, we can use the VNC Client to access the server and install Linux.

Note: In most of cases the VNC option is the best solution.

There are at least four methods of installation available to install your Linux distribution:

- ▶ Install from a hard disk or a DASD.
- ▶ Install via NFS.
- ▶ Install via FTP.
- ▶ Install via HTTP.

When we are using NFS, FTP, or HTTP methods, we need to fill in an additional screen configuration.

Note: For SuSE SLES 10 and RHEL 5.1 we can choose a Layer 2 connection to connect to NFS and the FTP server.

When we start the installation, we first make some configuration decisions before the packages become available to be installed. To install:

1. Language selection - Select the language that you would like to use during the installation.

Note: This choice is valid only for the installation process. To configure what language your system will supported you must configure it in the language support configuration panel.

2. Disk partitioning setup - At this option, we can choose automatic partitioning or manual disk partitioning. Here we configure the partition sizes used by Linux. See Figure 9-1.

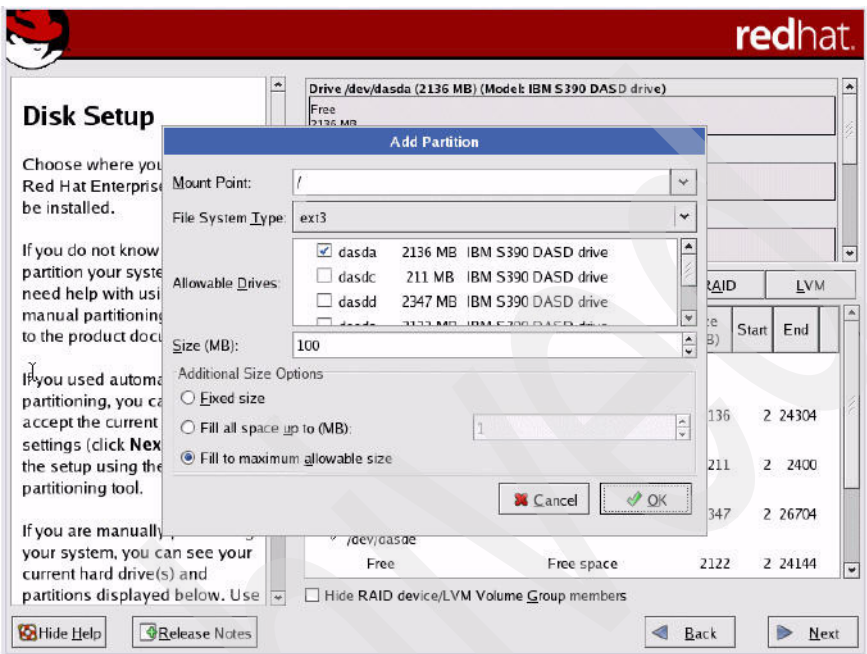
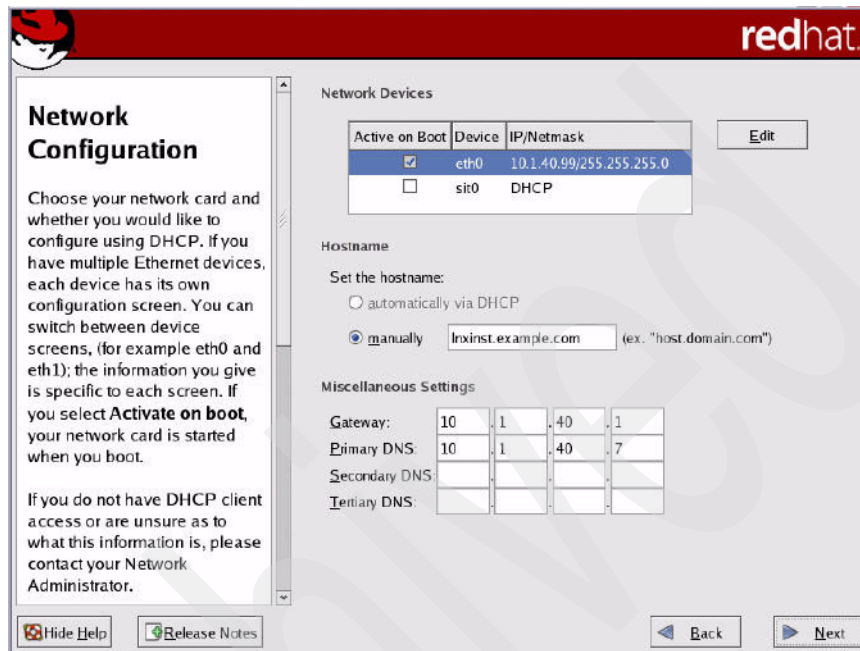


Figure 9-1 Adding partitions on Linux installation

Note: For a production server we highly recommend using the manual disk partitioning option.

3. Network configuration - Configure host name, gateway, and DNS. If we have multiple network cards, each device has its own configuration screen. See Figure 9-2.



The image shows a Red Hat Network Configuration window. The title bar is red with the Red Hat logo and the text "redhat.". The window is divided into several sections. On the left, there is a "Network Configuration" section with a heading and a paragraph of text. Below this, there are two buttons: "Hide Help" and "Release Notes". The main area of the window is titled "Network Devices" and contains a table with columns "Active on Boot", "Device", and "IP/Netmask". There is an "Edit" button to the right of the table. Below the table, there is a "Hostname" section with a heading and two radio buttons: "automatically via DHCP" and "manually". The "manually" radio button is selected, and there is a text input field with the value "lnxinst.example.com" and a hint "(ex. 'host.domain.com')". Below the hostname section, there is a "Miscellaneous Settings" section with a heading and four rows of settings: "Gateway:", "Primary DNS:", "Secondary DNS:", and "Tertiary DNS:". Each row has a table of four input fields. The "Gateway:" row has values "10", "1", "40", and "1". The "Primary DNS:" row has values "10", "1", "40", and "7". The "Secondary DNS:" and "Tertiary DNS:" rows are empty. At the bottom right of the window, there are two buttons: "Back" and "Next".

Network Configuration

Choose your network card and whether you would like to configure using DHCP. If you have multiple Ethernet devices, each device has its own configuration screen. You can switch between device screens, (for example eth0 and eth1); the information you give is specific to each screen. If you select **Activate on boot**, your network card is started when you boot.

If you do not have DHCP client access or are unsure as to what this information is, please contact your Network Administrator.

Network Devices

Active on Boot	Device	IP/Netmask
<input checked="" type="checkbox"/>	eth0	10.1.40.99/255.255.255.0
<input type="checkbox"/>	sit0	DHCP

Hostname

Set the hostname:

☐ automatically via DHCP

☒ manually (ex. "host.domain.com")

Miscellaneous Settings

Gateway:

Primary DNS:

Secondary DNS:

Tertiary DNS:

Figure 9-2 Configuring network

4. Firewall configuration - At this point, we can decide to prevent access to your computer. We can choose the No Firewall or Enable Firewall option. If Enable firewall is checked we can allow access to some services such as remote login (SSH), file transfer (FTP), mail server (SMTP), and Web server (HTTP/HTTPS). See Figure 9-3.

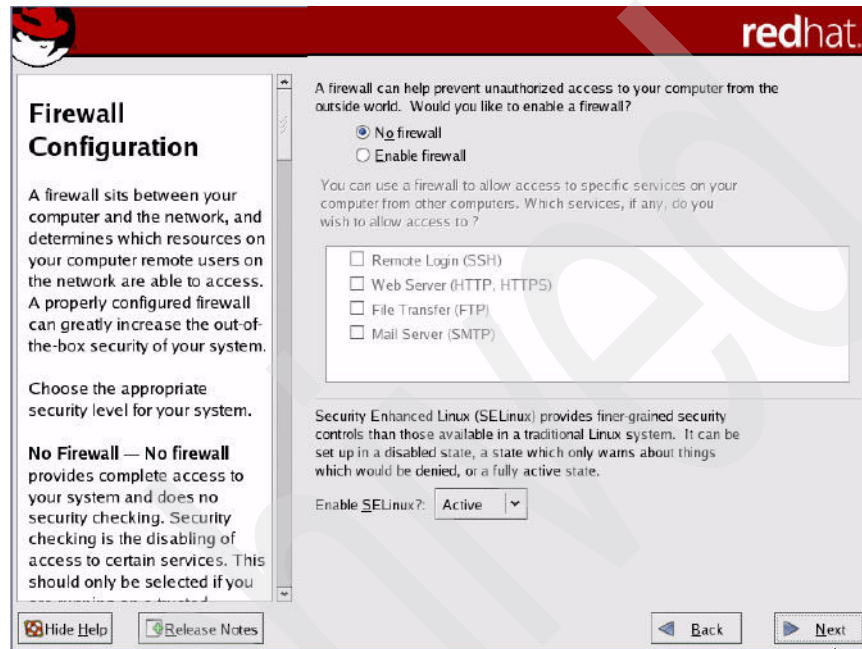


Figure 9-3 Firewall configuration

Note: If you prefer, you can enable/disable these services after the system is installed.

5. On following screens we can select a language to support, set the time zone and root password, and select packages that we want to install.

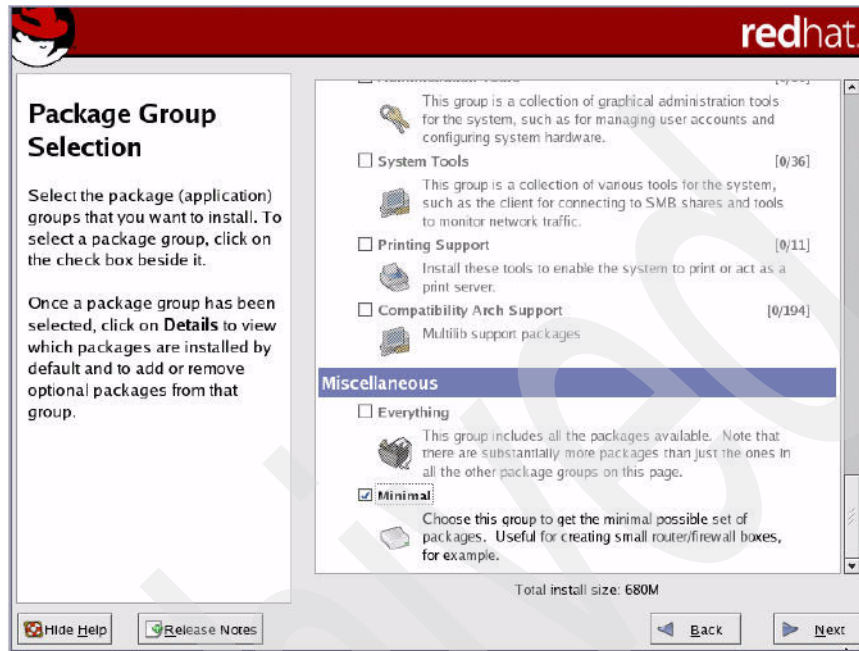


Figure 9-4 Package configuration

After this, Linux will be installed and ready to use.

Note: More information about the detailed Linux installation on S/390 architecture can be found in *IBM z/VM and Linux on IBM System z: Virtualization Cookbook for Red Hat Enterprise Linux 4*, SG24-7272, and at: <http://www.ibm.com/linux>

9.2 Gathering information regarding installation process

Note: This chapter was written based on Red Hat Enterprise Linux Version 5. We include comments when the same topic has a different file name or location for SUSE Linux Enterprise Server Version 10.

Gathering information provided during the installation

During the installation, all information is saved in the file `/root/install.log`, but we are unable to access this file once we reboot the system. In this file we encounter information about which packages and versions were installed based on your installation choice. See Example 9-1.

Example 9-1 Partial content of `/root/install.log`

```
Installing eel2-devel - 2.16.1-1.el5.s390x
Installing bug-buddy - 1:2.16.0-2.el5.s390x
Installing setroubleshoot - 1.8.11-4.el5.noarch
Installing evolution-data-server-devel - 1.8.0-15.el5.s390
Installing devhelp - 0.12-9.el5.s390
Installing gnome-desktop-devel - 2.16.0-1.fc6.s390
Installing gnome-python2-bonobo - 2.16.0-1.fc6.s390x
Installing sabayon - 2.12.4-3.el5.s390x
Installing eel2-devel - 2.16.1-1.el5.s390
Installing krb5-auth-dialog - 0.7-1.s390x
Installing gtkhtml3 - 3.12.0-1.fc6.s390x
Installing orca - 1.0.0-5.el5.s390x
Installing gnome-python2-gnomevfs - 2.16.0-1.fc6.s390x
Installing gnome-terminal - 2.16.0-3.el5.s390x
Installing libgnome-devel - 2.16.0-6.el5.s390
Installing system-config-date - 1.8.12-1.el5.noarch
Installing nautilus - 2.16.2-6.el5.s390x
Installing alacarte - 0.10.0-1.fc6.noarch
Installing yelp - 2.16.0-13.el5.s390x
Installing eog - 2.16.0.1-6.el5.s390x
Installing gnome-applets - 1:2.16.0.1-19.el5.s390x
Installing ImageMagick - 6.2.8.0-3.el5.4.s390
Installing gnome-system-monitor - 2.16.0-3.el5.s390x
Installing im-chooser - 0.3.3-6.el5.s390x
Installing nautilus-cd-burner - 2.16.0-7.el5.s390x
Installing evince - 0.6.0-8.el5.s390x
Installing nautilus-cd-burner - 2.16.0-7.el5.s390
Installing pirut - 1.2.10-1.el5.noarch
Installing file-roller - 2.16.0-2.fc6.s390x
Installing dogtail - 0.6.1-2.el5.noarch
Installing gnome-power-manager - 2.16.0-7.el5.s390x
Installing gnome-utils - 1:2.16.0-3.el5.s390
Installing gnome-session - 2.16.0-6.el5.s390x
```

Gathering information provided after the installation

After the installation, most of the log files are located in the `/var/log` directory. In this directory we can find system and applications default logs. An example of application logs shows us the `gdm` and `cup` directories that contain logs for the applications Gnome Display Manager and Common UNIX® Printing System, respectively. In Figure 9-5, we can verify that more logs are available in the `/var/log` directory.

```
lnx.itso.ibm.com:/var/log # ls
YaST2      faillog  mail.info  scpm        zmd-backend.log
apparmor   firewall mail.warn  slpd.log    zmd-messages.log
audit      gdm      messages  smpppd
boot.log   krb5     news      susehelp.log boot.msg    lastlog
ntp        warn
boot.omsg  mail     sa         wtmp
cups       mail.err samba      xinetd.log
lnx.itso.ibm.com#
```

Figure 9-5 Content of `/var/log` directory

Regarding gathering information, the daemon `syslogd` and the file `syslog.conf` are essential keys to understanding what is printed to the `/var/log/` files.

`Syslogd` is a daemon that reads and logs messages to the system console or log files. The `syslog.conf` is the configuration file to this daemon. At `syslog.conf` we can configure what kind of and where the information will be available.

Example 9-2 shows a `syslog.conf` file. Note that all information regarding notice, kern debug, and so on, will be recorded on `/var/log/messages`.

Example 9-2 `Syslog.conf`

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.* /dev/console
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none /var/log/messages

# The authpriv file has restricted access.
authpriv.* /var/log/secure

# Log all the mail messages in one place.
mail.* /var/log/maillog
```

```
# Log cron stuff
cron.* /var/log/cron

# Everybody gets emergency messages
*.emerg *

# Save news errors of level crit and higher in a special file.
uucp,news.crit /var/log/spooler

# Save boot messages also to boot.log
local7.* /var/log/boot.log
```

Note: For SUSE, we need to configure the `syslog-ng.conf` to customize the type of information that we want to see in `/var/log/messages`. The file `syslog-ng.conf` is located in the `/etc/syslog-ng` directory.

Analyzing the `syslog.conf` example, we note that the line `mail.* /var/log/maillog` is responsible for writing all information regarding mail in the file `/var/log/maillog`. See Example 9-3 to understand what and how the information is printed.

Example 9-3 Partial content of /var/log/maillog

```
Sep 13 15:40:34 linux postfix[3343]: warning:My hostname lnxmr2 is not
a fully qualified name - set myhostname or my domain in
/etc/postfix/main.cf
Sep 13 15:40:35 linux postfix/postfix-script: fatal: the Postfix mail
system is not running
Sep 13 15:41:38 linux postfix/postfix-script: fatal: the Postfix mail
system is not running
Sep 13 15:45:31 lnxmr2 postfix/postfix-script: starting the Postfix
mail system
Sep 13 15:45:31 lnxmr2 postfix/master[5627]: daemon started -- version
2.2.9, configuration /etc/postfix
Sep 13 15:47:43 lnxmr2 postfix/postfix-script: refreshing the Postfix
mail system
Sep 13 15:47:43 lnxmr2 postfix/master[5627]: reload configuration
/etc/postfix
Sep 13 17:55:02 lnxmr2 postfix/postfix-script: fatal: the Postfix mail
system is not running
```

The line `*.info;mail.none;authpriv.none;cron.none /var/log/messages` is responsible for not allowing private authentication messages on the file `/var/log/messages`.

See Example 9-4 to understand how the information configured in the file `/etc/syslog.conf` is recorded on `/var/log/messages`.

Example 9-4 Example of `/var/log/messages`

```
Mar 10 12:54:05 linux kernel: klogd 1.4.1, log source = /proc/kmsg
started.
Mar 10 12:54:05 linux kernel: AppArmor: AppArmor (version
2.0-19.43r6320) initialized
Mar 10 12:54:05 linux kernel: audit(1205168032.210:2): AppArmor
(version 2.0-19.43r6320) initialized
Mar 10 12:54:05 linux syslog-ng[1733]: Changing permissions on special
file /dev/xconsole
Mar 10 12:54:05 linux syslog-ng[1733]: Changing permissions on special
file /dev/tty10
Mar 10 12:54:05 linux kernel:
Mar 10 12:54:05 linux kernel: ip_tables: (C) 2000-2006 Netfilter Core
Team
Mar 10 12:54:06 linux kernel: SCSI subsystem initialized
Mar 10 12:54:10 linux kernel: IUCV lowlevel driver initialized
Mar 10 12:54:10 linux kernel: iucv: NETIUCV driver initialized
Mar 10 12:54:12 linux kernel: eth0: no IPv6 routers present
Mar 10 12:54:20 linux zmd: NetworkManagerModule (WARN): Failed to
connect to NetworkManager
Mar 10 12:54:27 linux zmd: Daemon (WARN): Not starting remote web
server
Mar 10 12:55:54 linux su: (to root) root on none
Mar 10 12:56:30 linux kernel: ip6_tables: (C) 2000-2006 Netfilter Core
Team
Mar 10 12:56:30 linux SuSEfirewall2: Warning: ip6tables does not
support state matching. Extended IPv6 support disabled.
Mar 10 12:56:30 linux SuSEfirewall2: Setting up rules from
/etc/sysconfig/SuSEfirewall2 ...
Mar 10 12:56:30 linux SuSEfirewall2: Warning: no interface active
Mar 10 12:56:30 linux kernel: Netfilter messages via NETLINK v0.30.
Mar 10 12:56:30 linux kernel: ip_conntrack version 2.4 (8192 buckets,
65536 max) - 296 bytes per conntrack
Mar 10 12:56:30 linux SuSEfirewall2: batch committing...
Mar 10 12:56:31 linux SuSEfirewall2: Firewall rules successfully set
Mar 10 12:56:32 linux SuSEfirewall2: Warning: ip6tables does not
support state matching. Extended IPv6 support disabled.
Mar 10 12:56:32 linux SuSEfirewall2: batch committing...
Mar 10 12:56:32 linux SuSEfirewall2: Firewall rules unloaded.
Mar 10 12:56:32 linux SuSEfirewall2: Warning: ip6tables does not
support state matching. Extended IPv6 support disabled.
```

```

Mar 10 12:56:32 linux SuSEfirewall2: Setting up rules from
/etc/sysconfig/SuSEfirewall2 ...
Mar 10 12:56:32 linux SuSEfirewall2: batch committing...
Mar 10 12:56:32 linux SuSEfirewall2: Firewall rules successfully set
Mar 10 12:57:40 linux SuSEfirewall2: Warning: ip6tables does not
support state matching. Extended IPv6 support disabled.
Mar 10 12:57:40 linux SuSEfirewall2: batch committing...
Mar 10 12:57:40 linux SuSEfirewall2: Firewall rules unloaded.
Mar 10 12:57:40 linux SuSEfirewall2: Warning: ip6tables does not
support state matching. Extended IPv6 support disabled.
Mar 10 12:57:40 linux SuSEfirewall2: Setting up rules from
/etc/sysconfig/SuSEfirewall2 ...
Mar 10 12:57:40 linux SuSEfirewall2: batch committing...
Mar 10 12:57:40 linux SuSEfirewall2: Firewall rules successfully set
Mar 10 12:59:07 linux syslog-ng[1733]: SIGHUP received, restarting
syslog-ng
Mar 10 12:59:08 lxorainst init: Re-reading inittab
Mar 10 12:59:08 lxorainst syslog-ng[1733]: new configuration
initialized
Mar 10 13:00:56 lxorainst ifdown:      eth0
Mar 10 13:00:56 lxorainst ifdown:      eth0      configuration:
qeth-bus-ccw-0.0.0650
Mar 10 13:00:56 lxinst init: Entering runlevel: 5
Mar 10 13:00:57 lxinst kernel: Kernel logging (proc) stopped.
Mar 10 13:00:57 lxinst kernel: Kernel log daemon terminating.
Mar 10 13:00:57 lxinst ifup:      lo
Mar 10 13:00:57 lxinst ifup:      lo
Mar 10 13:00:57 lxinst ifup: IP address: 127.0.0.1/8
Mar 10 13:00:57 lxinst ifup:
Mar 10 13:00:57 lxinst ifup:      eth0

```

For a better understanding of this we discuss parts of `/var/log/messages` logs in the following sections.

Information regarding applications (for example, SuSEfirewall)

This part of the log shows information about the SuSEfirewall application. We can note that there is information about firewall rules status process, warnings, and so on. All information printed here is an execution result from the SuSEfirewall2 configuration file.

Example 9-5 SuSEfirewall logs on /var/log/messages

```

Mar 10 12:56:30 linux SuSEfirewall2: Warning: ip6tables does not
support state matching. Extended IPv6 support disabled.

```



```
Mar 10 12:56:30 linux SuSEfirewall2: Setting up rules from
/etc/sysconfig/SuSEfirewall2 ...
Mar 10 12:56:30 linux SuSEfirewall2: Warning: no interface active
Mar 10 12:56:30 linux kernel: Netfilter messages via NETLINK v0.30.
Mar 10 12:56:30 linux kernel: ip_conntrack version 2.4 (8192 buckets,
65536 max) - 296 bytes per conntrack
Mar 10 12:56:30 linux SuSEfirewall2: batch committing...
Mar 10 12:56:31 linux SuSEfirewall2: Firewall rules successfully set
Mar 10 12:56:32 linux SuSEfirewall2: Warning: ip6tables does not
support state matching. Extended IPv6 support disabled.
Mar 10 12:56:32 linux SuSEfirewall2: batch committing...
Mar 10 12:56:32 linux SuSEfirewall2: Firewall rules unloaded.
Mar 10 12:56:32 linux SuSEfirewall2: Warning: ip6tables does not
support state matching. Extended IPv6 support disabled.
Mar 10 12:56:32 linux SuSEfirewall2: Setting up rules from
/etc/sysconfig/SuSEfirewall2 ...
Mar 10 12:56:32 linux SuSEfirewall2: batch committing...
Mar 10 12:56:32 linux SuSEfirewall2: Firewall rules successfully set
Mar 10 12:57:40 linux SuSEfirewall2: Warning: ip6tables does not
support state matching. Extended IPv6 support disabled.
Mar 10 12:57:40 linux SuSEfirewall2: batch committing...
Mar 10 12:57:40 linux SuSEfirewall2: Firewall rules unloaded.
Mar 10 12:57:40 linux SuSEfirewall2: Warning: ip6tables does not
support state matching. Extended IPv6 support disabled.
Mar 10 12:57:40 linux SuSEfirewall2: Setting up rules from
/etc/sysconfig/SuSEfirewall2 ...
Mar 10 12:57:40 linux SuSEfirewall2: batch committing...
Mar 10 12:57:40 linux SuSEfirewall2: Firewall rules successfully set
```

Information regarding Linux kernel

This part of the log shows information about the kernel. This information is printed during the Linux boot.

Example 9-6 Kernel logs on /var/log/messages

```
Mar 10 12:54:05 linux kernel: klogd 1.4.1, log source = /proc/kmsg
started.
Mar 10 12:54:05 linux kernel: AppArmor: AppArmor (version
2.0-19.43r6320) initialized
Mar 10 12:54:05 linux kernel: audit(1205168032.210:2): AppArmor
(version 2.0-19.43r6320) initialized
Mar 10 12:54:05 linux kernel:
Mar 10 12:54:05 linux kernel: ip_tables: (C) 2000-2006 Netfilter Core
Team
Mar 10 12:54:06 linux kernel: SCSI subsystem initialized
```

```
Mar 10 12:54:10 linux kernel: IUCV lowlevel driver initialized
Mar 10 12:54:10 linux kernel: iucv: NETIUCV driver initialized
Mar 10 12:54:12 linux kernel: eth0: no IPv6 routers present
```

Information regarding network cards

This part of the log (Example 9-7) shows information regarding network cards. This information is recorded while the network card is starting. If we have more than one network card, all information devices will be printed.

Example 9-7 Network cards logs on /var/log/messages

```
Mar 10 13:00:57 lxinst ifup:      lo
Mar 10 13:00:57 lxinst ifup:      lo
Mar 10 13:00:57 lxinst ifup: IP address: 127.0.0.1/8
Mar 10 13:00:57 lxinst ifup:
Mar 10 13:00:57 lxinst ifup:      eth0
```

9.3 Scenarios

In this section, we cover common and possible problems that could occur during the installation or device configurations. When we are talking about Installation problems, we can say that we have two possibilities:

- Problems during the installation

These are problems that occur while running the program interface installer, for example, a problem with creating partitions, packages dependencies, devices not found, and problems with network setup.

- Problems after the installation

These are problems that occur after the Linux install is done successfully, for example, problems with device configurations and problems logging into Linux.

9.3.1 Trouble during the installation

In this section we discuss problems during the installation.

Scenario 1: Problems cloning Linux system when SuSE v10 and SP1 are used

Problem determination: If we take the default device names of SP1 during installation, your system will be unable to clone.

Solution: We can change device naming through YaST or change it directly in `/etc/fstab`.

```
lnxihs:~ # cat /etc/fstab

/dev/dasda1 / ext3 acl,user_xattr 1 1
/dev/dasdb1 swap swap defaults 0 0
proc /proc proc defaults 0 0
sysfs /sys sysfs noauto 0 0
debugfs /sys/kernel/debug debugfs noauto 0 0
devpts /dev/pts devpts mode=0620,gid=5 0 0
```

Figure 9-6 `/etc/fstab` example

9.3.2 Trouble after the installation

In this section we discuss problems after the installation.

Scenario 1: forgotten root password (SuSE or Red Hat)

Problem description: We have configured the root password during the installation but when we try to log in after the reboot we have problems.

This problem can occur when we forget the password or typed it in wrong during the installation.

Solution: To solve this problem, boot your Linux system using Linux single mode or init 1. When are you booting at Linux single mode or init 1 the root file system is mounted as read-only. After the boot, we will have access to the prompt, so we must type the command below:

```
# passwd root
```

We will set the new root password. Reboot your system using init 6 or by typing the command below:

```
# shutdown -r now
```


Booting problems

This chapter helps you to identify most problems that you could encounter with the boot process. In the first section we show how the boot process normally works. This makes the problem determination much easier.

This chapter covers the following topics:

- ▶ Understanding the boot process
- ▶ Gathering information regarding boot
- ▶ Commands
- ▶ Boot problem examples
- ▶ Scenarios

10.1 Understanding the boot process

The Linux operating system for System z has four phases on its boot process. They are IPL, boot loader, kernel, and Init phases. See Figure 10-1.

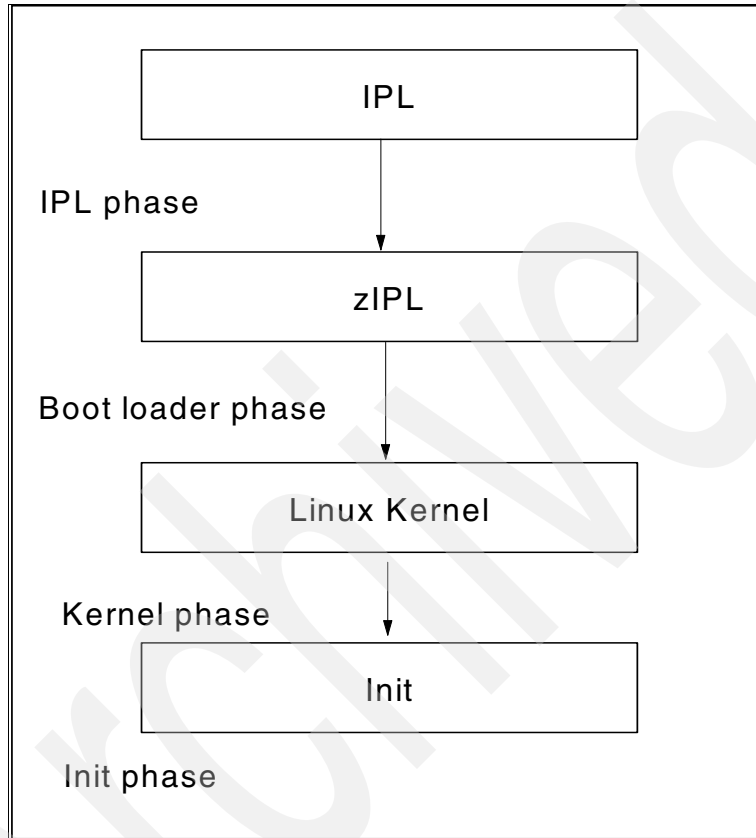


Figure 10-1 Boot process

Now we describe each process phase.

10.1.1 IPL phase

Initial Program Loader (IPL) is a mainframe term for the loading of the operating system into the computer's main memory. IPL is to mainframe as boot is for other architectures. See Figure 10-2.

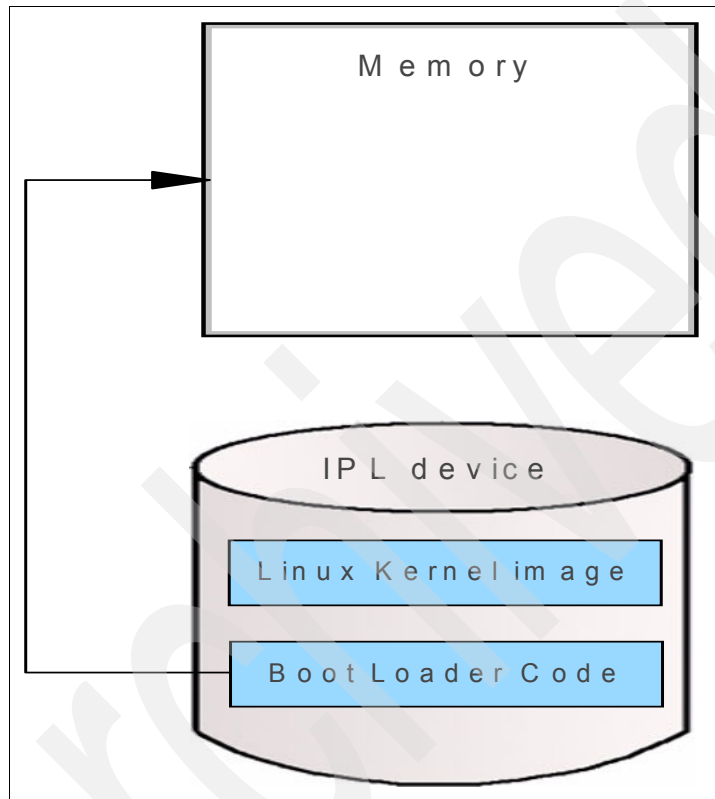


Figure 10-2 IPL phase.

The IPL phase consists of reading the boot loader code from the operator-specified device into memory and starting boot loader (zIPL).

DASD, SCSI disk, tape, and VM reader can be used as IPL devices.

10.1.2 Boot loader phase

The main boot loader function is to load other software for the operating system to start. It is found at the start of the IPL device. The boot loader used for IBM S/390 and zSeries machines is called zIPL. Boot loader code is made by the

`/sbin/zipl` command and installed on a user-specified IPL device. Figure 10-3 shows an example of how zIPL works.

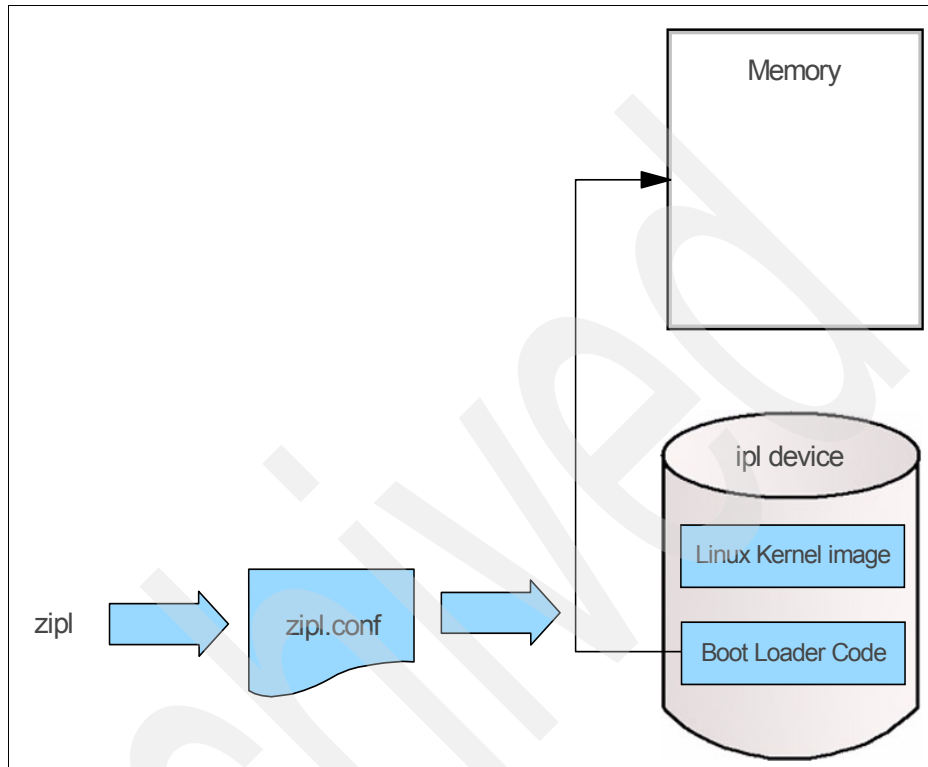


Figure 10-3 How zIPL works

The bootloader phase consists of loading the initial RAM disk image (initrd) and starting the Linux kernel, and then finally booting the Linux Operating System.

After loading boot Linux kernel, the control is passed to the Linux kernel, as shown in Figure 10-4.

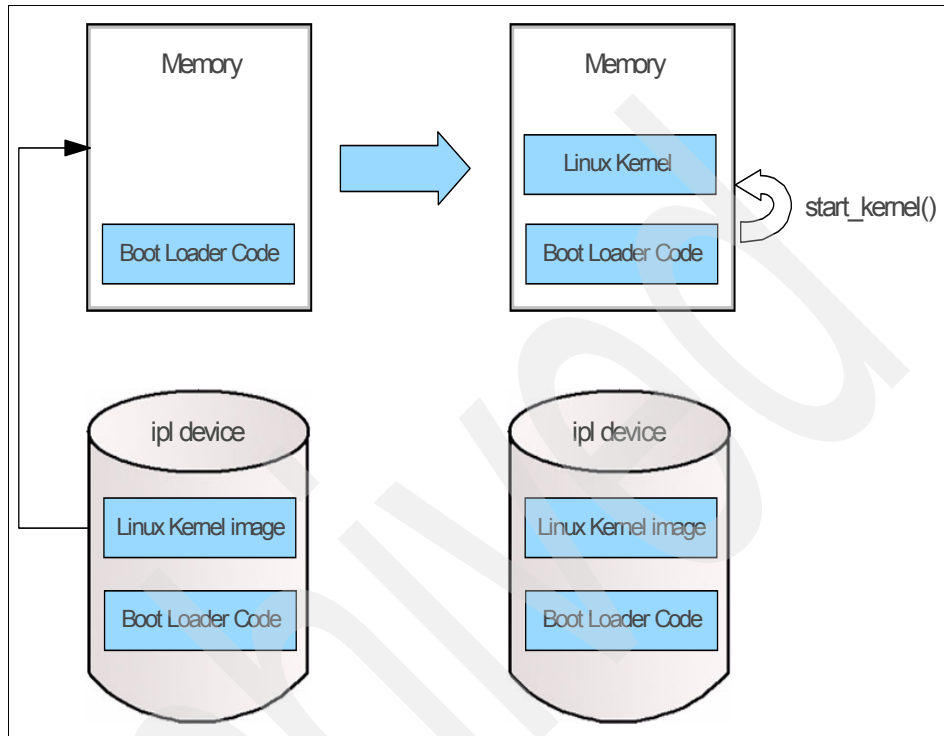


Figure 10-4 Boot loader phase

You will find a configuration for a zIPL. See Example 10-1.

Example 10-1 Configuration for a zIPL

```
[defaultboot]
defaultmenu = menu

:menu
  default = 1
  prompt = 1
  target = /boot/zipl
  timeout = 1
  1 = ipl

[ipl]
  image = /boot/image
  target = /boot/zipl
```

```
ramdisk = /boot/initrd,0x1000000  
parameters =  
"root=/dev/disk/by-id/ccw-IBM.7500000000BALB1.89e9.10-part1  
elevator=deadline TERM=dumb"
```

10.1.3 Kernel phase

Kernel is the heart of the operating system. The kernel is the heart of the operating system. Some functions of the kernel are to manage system resources (memory, CPU, and so on) and communicate between software and hardware.

The kernel phase consists of probing and initializing the hardware, as well as initializing the kernel subsystems and decompressing initrd and loading some device drivers. After that, mount the root file system with read-only access and check the file system with the fsck program, then start /sbin/init. After this the kernel will sleep, as shown in Figure 10-5.

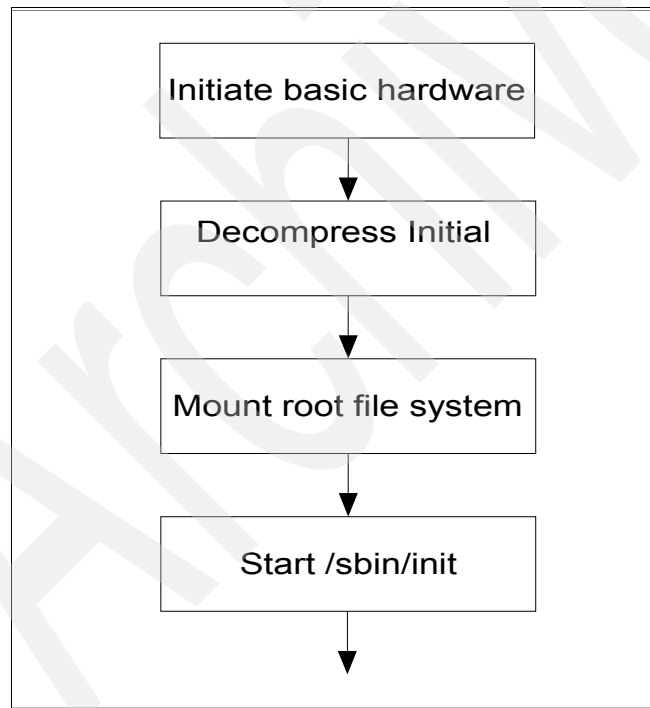


Figure 10-5 Kernel phase

Let us talk a little bit more about each kernel phase.

► Recognition and initialization of basic hardware

This step recognizes and initiates basic hardware (processors, memory, console, and so on) included in the kernel. Initialization of various kernel subsystems like SMP, page table, and NET are part of this step.

First of all, Linux kernel decompresses the kernel image on memory, then initializes several devices and subsystems with the `start_kernel()` function.

Devices and subsystems initialized by the kernel are:

- Initialize kernel page table.
- Outputs kernel banner.
- Initialize physical memory.
- Initialize CPU cache L1/L2.
- Initialize scheduler.
- Registration of interrupt, trap, and task gate.
- Initialize read-copy update (RCU).
- Initialize PID hash table.
- Initialize class-based kernel resource management (CKRM).
- Initialize timer.
- Initialize system clock.
- Initialize console.
- Initialize VFS.
- Initialize signal.

The process of PID 1 is started when the initialization of the main hardware is completed, and the following processing is executed by the `init ()` function.

Note: PID 1 is called init process and it stays alive until Linux is shut down.

- Initialize multi-processor.
- Initialize kernel addition function.
 - Initialize devices management data.
 - Initialize network management data.
- Process Initial RAM DISK (`initrd`).
- Mount / file system.
- Run `/sbin/init`.

► Decompressing initial RAM disk (`initrd`)

This process is required to mount the root file system, load kernel drivers modules, and recognize devices that should be recognized when each service starts.

RAM disk image file includes driver, basic commands and scripts (linuxrc) to mount real root file system. After the mount, the linuxrc script is executed and the following processes occur:

- Processing of linuxrc script
- Mounting the proc and sys file systems
- Starting the udev service
- Loading the driver module /lib/modules or its subdirectory
- Recognizing DASD devices and making them online
- Recognizing FCP channels and putting FCP volumes online
- Recognizing LVM volumes and activating them
- Recognizing real root file systems
- Writing path information to /proc/sys/kernel/real-root-dev

The mount preparation of the root file system is completed by processing the linuxrc script.

► Mounting root file system

This step mounts the root file system in read-only mode. Note that read-write (rw) mount is done later by the init phase. At this point all file systems are checked with the **fsck** command.

Linux kernel mounts the root file system with read-only (ro) mode, and **fsck** is executed. Then the device that includes the real root file system can be recognized. At this time, initrd that was mounted previously as a root file system is saved in the /initrd directory.

Note: In SLES9 and SLES10, the initrd is annulled because /initrd does not exist.

The reason for doing the mount with read-only is to prevent the root file system from breaking and being changed by fsck.

As for the root file system, remount is done with read-write (rw) in the init phase.

► Starting /sbin/init

At this point Linux kernel will sleep. The /sbin/init and the child process do all the start processing afterwards.

10.1.4 Init phase

The init phase is the process control initialization. Init is the parent of all process. Its primary role is to create a process from a script stored in the file `/etc/inittab`. This file usually has entries that cause init to spawn gettys on each line that users can log in. It also controls autonomous processes required by any particular system.

The `/sbin/init` controls the following processes:

- ▶ Look at the specific runlevel setting as specified in the runlevels configuration, set the source function library for the system, and start applicable processes (following `/etc/inittab`).
- ▶ Run `/etc/init.d/boot.*` (recognize LVM volumes, mount all file systems, and so on).
- ▶ Start the `/etc/init.d/rc[0-6].d/*` daemon.

Let us go into more detail about Init phase.

`/sbin/init` is executed when initialization by the kernel is completed, and the init process is generated.

The primary role of initialization by the init process is to create processes from a configuration script stored in the file `/etc/inittab`.

Various settings, initialization, and start processing are processed following `/etc/inittab`.

- ▶ Runlevel setting
 - Usually this is set by the value of init default in `/etc/inittab`.
 - The specified runlevel is recorded in `/var/run/utmp`, and referred by `/sbin/runlevel`.
- ▶ Running the `/etc/init.d/boot`
 - Common system initialization that does not depend on runlevel
 - Time setting, initializing swap device, checking and mounting file systems, initializing the loop back device, initializing quota, initializing the `/proc` file system, and so on, are done here.
- ▶ Running service scripts
 - Starting various demons according to runlevel: `/etc/init.d/rcX.d/*` scripts (X is the runlevel).
 - The configuration file to which each service script refers is stored in `/etc/sysconfig/`.

- The starting service of each runlevel is set by the **chkconfig** command and the **insserv** command.
- ▶ Starting getty
 - After starting all the daemons, getty is started.

Now we describe some parts of the `/etc/inittab` configuration file.

- ▶ The `/etc/inittab` format is `id:runlevels:action:process`.
- ▶ Some actions that the **init** command take when the process starts are:
 - `bootwait` - The process is executed during system boot.
 - `initdefault` - The default runlevel is configured here when the system starts.
 - `once` - The process is started once when the specified runlevel is entered.
- ▶ The process starts only once when booting.
 - `wait` - The process will be started once when the specified runlevel is entered and `init` will wait for its termination.
 - `respawn` - The process will be restarted whenever it terminates.

See the `/etc/inittab/` default configuration in Example 10-2.

Example 10-2 Default configuration of `/etc/inittab`

```
# The default runlevel is defined here
id:5:initdefault:

# First script to be executed, if not booting in emergency (-b) mode
si::bootwait:/etc/init.d/boot

# /etc/init.d/rc takes care of runlevel handling
#
# runlevel 0 is System halt (Do not use this for initdefault!)
# runlevel 1 is Single user mode
# runlevel 2 is Local multiuser without remote network (e.g. NFS)
# runlevel 3 is Full multiuser with network
# runlevel 4 is Not used
# runlevel 5 is Full multiuser with network and xdm
# runlevel 6 is System reboot (Do not use this for initdefault!)
#
10:0:wait:/etc/init.d/rc 0
11:1:wait:/etc/init.d/rc 1
12:2:wait:/etc/init.d/rc 2
13:3:wait:/etc/init.d/rc 3
#14:4:wait:/etc/init.d/rc 4
15:5:wait:/etc/init.d/rc 5
```

```

16:6:wait:/etc/init.d/rc 6

# what to do in single-user mode
ls:S:wait:/etc/init.d/rc S
~~:S:respawn:/sbin/sulogin

# Default HMC/3215/3270 console:
1:2345:respawn:/sbin/mingetty --noclear /dev/ttyS0 dumb
# VT220(Linux) console:
#2:2345:respawn:/sbin/mingetty --noclear /dev/ttyS1 xterm
# additional 3270 terminals:
#3:2345:respawn:/sbin/mingetty --noclear /dev/3270/ttycons dumb

# what to do when CTRL-ALT-DEL is pressed
ca::ctrlaltdel:/sbin/shutdown -r -t 4 now

# not used for now:
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
#pn::powerfail:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop
sh:12345:powerfail:/sbin/shutdown -h now THE POWER IS FAILING

```

/etc/init.d/boot script

The script /etc/init.d/boot is used to:

- ▶ Activate SELinux.
- ▶ Mount the /proc file system.
- ▶ Mount the /sys file system.
- ▶ Mount /dev/pts.
- ▶ Start blogd.
- ▶ Execute various start scripts (these scripts are located at /etc/init.d/boot.d/).
- ▶ Execute the /etc/init.d/boot.local script.
- ▶ Stop blogd.

Executing /etc/init.d/boot scripts

The scripts placed on /etc/init.d/boot.d/ are called and executed by the /etc/init.d/boot script. The following processes are executed by them:

- ▶ Loading kernel modules
- ▶ Activating swap file systems
- ▶ Activating SUSE Configuration Profile Management (SCPM)
- ▶ Updating the library path

- ▶ Setting the host name
- ▶ Activating the loop back device and mounting it
- ▶ Setting the time slice of scheduler
- ▶ Setting the IP setting (parameter settings related to N/W such as IP forwarding)
- ▶ Setting the kernel parameter setting specified in `etc/sysctl.conf`

Note: At `/etc/init.d/boot.d`, processes that start with `Sxxx` (for example, `S12SusSEfirewall2_init`) start when the machine boots, while processes that start with `Kxxx` (for example, `K20boot.rootfsck`) are used to kill when the machine boots or is powered off.

Starting Getty

When the service start process has finished, the `getty` program is started by the `init` process, and the login prompt is displayed. As a result, login of the system becomes possible.

10.2 Where to gather information regarding boot process

There are many ways to collect information that are provided by the system during the boot. You can collect information by accessing the log files `/var/log/boot.msg`, `/var/log/oboot.msg`, and `/var/log/messages`. Another way to get information is using the `dmesg` command. See Figure 10-6 on page 297.

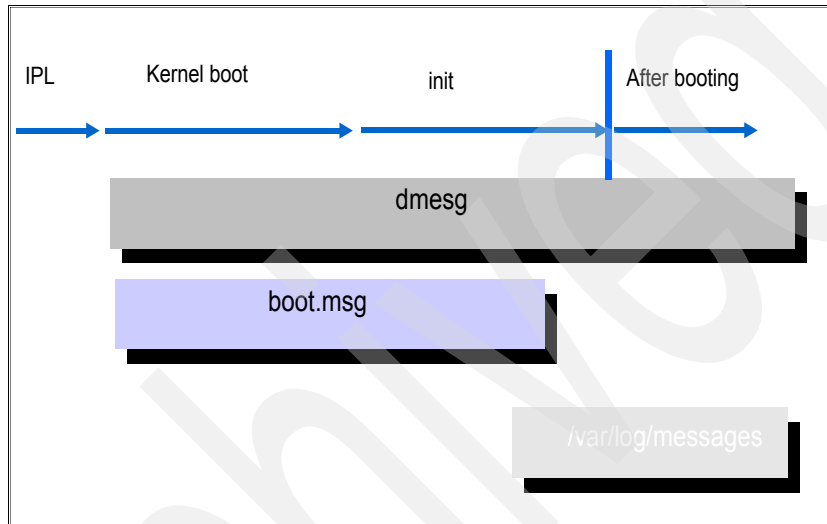


Figure 10-6 Gathering information.

The booting problem distinction is done based on the boot message.

During the boot we can check the log output destination in two environments:

- ▶ LPAR
 - Operating system messages of HMC (default console)
 - Integrated ASCII Console (configuration is needed)
- ▶ Under VM
 - Personal Communications 3270 (P-COMM) Screen Log
 - Integrated 3270 Console of HMC

After the boot we can check the output destination:

- ▶ `dmesg` command
- ▶ `/var/log/boot.msg` file
- ▶ `/var/log/boot.omsf` file
- ▶ `/var/log/messages`

Note: During problem determination, if the system does not respond, we can do a message dump by Magic System Request Key (Magic SysRq key). This sends a request to the kernel directly and displays a message. The output is displayed in the console and /var/log/message.

Below is more information about these options:

- ▶ The **dmesg** command - diagnostic message. **dmesg** prints out information provided by the kernel, but there is no information about IPL logs.
- ▶ The file /var/log/boot.msg contain information provided at the boot. The /var/log/boot.msg file contains information provided at boot time. During the initial phase, /dev/console is redirected to this file by the daemon named blogd.
- ▶ The file /var/log/oboot.msg contains information about the last boot.
- ▶ The file /var/log/messages contains information that is printed out by the daemon syslogd.

Example 10-3 shows the boot message to boot loader and kernel phases.

Example 10-3 boot loader and kernel phases messages boot

```
00: Booting default (ipl)...
Linux version 2.6.5-7.244-s390x (geeko@buildhost) (gcc version 3.3.3
(SuSE Linux))
#1 SMP Mon Dec 12 18:32:25 UTC 2005
We are running under VM (64 bit mode)
On node 0 totalpages: 524288
  DMA zone: 524288 pages, LIFO batch:31
  Normal zone: 0 pages, LIFO batch:1
  HighMem zone: 0 pages, LIFO batch:1
Built 1 zonelists
Kernel command line:
cio_ignore=all,!0.0.0009,!0.0.05A0-0.0.05A2,!0.0.3301-0.0.3
304_dasd=3301-3304 root=/dev/rootvg/lvroot selinux=0 TERM=dumb
elevator=cfq BOOT_IMAGE=0
PID hash table entries: 4096 (order 12: 65536 bytes)
CKRM initialization
..... initializing ClassType<taskclass> .....
..... initializing ClassType<socketclass> .....
CKRM initialization done
Dentry cache hash table entries: 524288 (order: 10, 4194304 bytes)
Inode-cache hash table entries: 262144 (order: 9, 2097152 bytes)
Memory: 2047744k/2097152k available (3456k kernel code, 0k reserved,
1075k data, 116k init)
```

Security Scaffold v1.0.0 initialized
SELinux: Disabled at boot.
Mount-cache hash table entries: 256 (order: 0, 4096 bytes)

To a better understand this, we split the message boot logs according the topics below:

- ▶ Starting kernel (boot loader phase)
Linux Version 2.6.5-7.244-s390x (geeko@buildhost) (gcc Version 3.3.3 (SuSE Linux)).
- ▶ Displaying starting environment (kernel boot phase -start_kernel())
We are running under VM (64 bit mode).
- ▶ Kernel parameter (kernel boot phase -start_kernel())
Kernel command line:
cio_ignore=all,!0.0.0009,!0.0.05A0-0.0.05A2,!0.0.3301-0.0.3304 dasd=3301-3304 root=/dev/rootvg/lvroot selinux=0 TERM=dumb elevator=cfq BOOT_IMAGE=0
- ▶ Memory initialization (kernel boot phase -start_kernel())
Memory: 2047744k/2097152k available (3456k kernel code, 0k reserved, 1075k data, 116k init).

Example 10-4 shows the boot message from the kernel phase.

Example 10-4 Kernel phase boot message

Starting udev
Creating devices
Loading kernel/fs/jbd/jbd.ko
Loading kernel/fs/ext3/ext3.ko
Loading kernel/drivers/md/dm-mod.ko
device-mapper: Allocated new minor_bits array for 1024 devices
device-mapper: 4.4.0-ioct1 (2005-01-12) initialised:
dm-devel@redhat.com
Loading kernel/drivers/md/dm-snapshot.ko
Loading kernel/drivers/s390/block/dasd_mod.ko dasd=3301-3304
Loading kernel/drivers/s390/block/dasd_eckd_mod.ko
dasd(eckd): 0.0.3301: 3390/0A(CU:3990/01) Cyl:3339 Head:15 Sec:224
Using cfq io scheduler
dasd(eckd): 0.0.3301: (4kB blks): 2404080kB at 48kB/trk compatible disk layout
dasda:VOL1/ 0X3301: dasda1 dasda2 dasda3
dasd(eckd): 0.0.3302: 3390/0A(CU:3990/01) Cyl:3339 Head:15 Sec:224

```

dasd(eckd): 0.0.3302: (4kB blks): 2404080kB at 48kB/trk compatible disk
layout
  dasdb:VOL1/ 0X3302: dasdb1
dasd(eckd): 0.0.3303: 3390/0A(CU:3990/01) Cyl:3339 Head:15 Sec:224
dasd(eckd): 0.0.3303: (4kB blks): 2404080kB at 48kB/trk compatible disk
layout
  dasdc:VOL1/ 0X3303: dasdc1
dasd(eckd): 0.0.3304: 3390/0A(CU:3990/01) Cyl:3339 Head:15 Sec:224
dasd(eckd): 0.0.3304: (4kB blks): 2404080kB at 48kB/trk compatible disk
layout
  dasdd:VOL1/ 0X3304: dasdd1
Waiting for /dev/mapper/control to appear: . ok
  Reading all physical volumes. This may take a while...
  Found volume group "rootvg" using metadata type lvm2
  1 logical volume(s) in volume group "rootvg" now active
kjournald starting. Commit interval 5 seconds
EXT3-fs: mounted filesystem with ordered data mode.
VFS: Mounted root (ext3 filesystem) readonly.
Trying to move old root to /initrd ... /initrd does not exist. Ignored.
Unmounting old root
Trying to free ramdisk memory ... failed
Freeing unused kernel memory: 116k freed

```

To a better understand this we split the message boot logs according to the topics below:

- ▶ **Loading device driver (kernel boot phase - processing initrd)**

```

Loading kernel/fs/jbd/jbd.ko
Loading kernel/fs/ext3/ext3.ko
Loading kernel/drivers/md/dm-mod.ko
Loading kernel/drivers/md/dm-snapshot.ko
Loading kernel/drivers/s390/block/dasd_mod.ko dasd=3301-3304
Loading kernel/drivers/s390/block/dasd_eckd_mod.ko

```
- ▶ **DASD recognition (kernel boot phase - processing initrd)**

```

dasd(eckd): 0.0.3301: 3390/0A(CU:3990/01) Cyl:3339 Head:15 Sec:224
Using cfq io scheduler
dasd(eckd): 0.0.3301: (4kB blks): 2404080kB at 48kB/trk compatible
disk layout
  dasda:VOL1/ 0X3301: dasda1 dasda2 dasda3
dasd(eckd): 0.0.3302: 3390/0A(CU:3990/01) Cyl:3339 Head:15 Sec:224
dasd(eckd): 0.0.3302: (4kB blks): 2404080kB at 48kB/trk compatible
disk layout
  dasdb:VOL1/ 0X3302: dasdb1
dasd(eckd): 0.0.3303: 3390/0A(CU:3990/01) Cyl:3339 Head:15 Sec:224

```

```

dasd(eckd): 0.0.3303: (4kB blks): 2404080kB at 48kB/trk compatible
disk layout
  dasdc:VOL1/ 0X3303: dasdc1
dasd(eckd): 0.0.3304: 3390/0A(CU:3990/01) Cyl:3339 Head:15 Sec:224
dasd(eckd): 0.0.3304: (4kB blks): 2404080kB at 48kB/trk compatible
disk layout
  dasdd:VOL1/ 0X3304: dasdd1

```

- Mounting/read-only (kernel boot phase - processing initrd)
 - VFS: Mounted root (ext3 filesystem) readonly.

At Example 10-5, see boot messages to the init phase.

Example 10-5 init phase boot message

```

INIT: version 2.85 booting
System Boot Control: Running /etc/init.d/boot
Mounting /proc filesystem..done
Mounting sysfs on /sys..done
Mounting /dev/pts..done
Boot logging started on /dev/ttyS0(/dev/console) at Wed May 2 09:18:50
2007
Mounting shared memory FS on /dev/shm..done
Activating swap-devices in /etc/fstab...
Adding 524872k swap on /dev/dasda2. Priority:42 extents:1
?1A..doneChecking root file system...
fsck 1.38 (30-Jun-2005)
?/sbin/fsck.ext3 (1) -- /dev/shm/root? fsck.ext3 -a /dev/shm/root
/dev/shm/root: clean, 84523/814400 files, 1091827/1626112 blocks
?1A..doneEXT3 FS on dm-0, internal journal
Hotplug is already active (disable with NOHOTPLUG=1 at the boot
prompt)..done
Scanning SCSI devices and filling /dev/scsi/ SCSI subsystem initialized
osst :I: Tape driver with OnStream support version 0.99.1
osst :I: $Id: osst.c,v 1.70 2003/12/23 14:22:12 wriede Exp $
st: Version 20040318, fixed bufsize 32768, s/g segs 256
..done
md: Autodetecting RAID arrays.
md: autorun ...
md: ... autorun DONE.
Activating device mapper...
Scanning for LVM volume groups...
  Reading all physical volumes. This may take a while...
  Found volume group "rootvg" using metadata type lvm2
Activating LVM volume groups...
  1 logical volume(s) in volume group "rootvg" now active

```

```

..done
Checking file systems...
fsck 1.38 (30-Jun-2005)
Checking all file systems.
?/sbin/fsck.ext3 (1) -- /boot? fsck.ext3 -a /dev/dasda1
/dev/dasda1: clean, 25/25728 files, 8784/25716 blocks
?/sbin/fsck.ext3 (1) -- /opt? fsck.ext3 -a /dev/dasddl
/dev/dasddl: clean, 2663/300960 files, 327754/600996 blocks
?1A..doneJBD: barrier-based sync failed on dm-0 - disabling barriers
Setting up..done
Mounting local file systems...
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
tmpfs on /dev/shm type tmpfs (rw)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
kjournald starting. Commit interval 5 seconds
EXT3 FS on dasda1, internal journal
EXT3-fs: mounted filesystem with ordered data mode.
/dev/dasda1 on /boot type ext3 (rw,acl,user_xattr)
mount: /dev/dasddl already mounted or /opt busy
?1A..failedLoading required kernel modules
?1A..doneSetting up the system clock..done
Activating remaining swap-devices in /etc/fstab...
?1A..doneRestore device permissions..done
Creating /var/log/boot.msg
?1A..doneSetting up timezone data..done
Setting scheduling timeslices ..unused
Setting up hostname 'sles9x-tak'..done
Setting up loopback interface  lo
    lo      IP address: 127.0.0.1/8
..done
Enabling syn flood protection..done
Disabling IP forwarding..done
..done
System Boot Control: The system has been set up
Skipped features:  ?80C ?17Dboot.sched
System Boot Control: Running /etc/init.d/boot.local
?1A..doneJBD: barrier-based sync failed on dasda1 - disabling barriers
INIT: Entering runlevel: 3
Boot logging started on /dev/ttyS0(/dev/console) at Wed May  2 09:21:40
2007
coldplug scanning ccw: ****..done
scanning input: ..done
Starting syslog services..done
Starting hardware scan on bootStarting CRON daemon..done

```

```
Starting Name Service Cache Daemon..done
..done
Master Resource Control: runlevel 3 has been reached
...
Welcome to SUSE LINUX Enterprise Server 9 (s390x) - Kernel
2.6.5-7.244-s390x (ttyS0).
itsolinux login:
```

To a better understand this we split the message boot logs according the topics below:

► **Mounting sysfs**

```
INIT: version 2.85 booting
System Boot Control: Running /etc/init.d/boot
Mounting /proc filesystem..done
Mounting sysfs on /sys..done
Mounting /dev/pts..done
```

► **Activating swap**

```
Activating swap-devices in /etc/fstab...
Adding 524872k swap on /dev/dasda2. Priority:42 extents:1
```

► **SCSI devices organization**

```
Scanning SCSI devices and filling /dev/scsi/ SCSI subsystem
initialized
osst :I: Tape driver with OnStream support version 0.99.1
osst :I: $Id: osst.c,v 1.70 2003/12/23 14:22:12 wriede Exp $
st: Version 20040318, fixed bufsize 32768, s/g segs 256
..done
```

► **Activating LVM**

```
Scanning for LVM volume groups...
Reading all physical volumes. This may take a while...
Found volume group "rootvg" using metadata type lvm2
Activating LVM volume groups...
1 logical volume(s) in volume group "rootvg" now active
..done
```

► **Running FSCK**

```
Checking file systems...
fsck 1.38 (30-Jun-2005)
Checking all file systems.
?/sbin/fsck.ext3 (1) -- /boot? fsck.ext3 -a /dev/dasda1
/dev/dasda1: clean, 25/25728 files, 8784/25716 blocks
```

```
?/sbin/fsck.ext3 (1) -- /opt? fsck.ext3 -a /dev/dasdd1
/dev/dasdd1: clean, 2663/300960 files, 327754/600996 blocks
?1A..doneJBD: barrier-based sync failed on dm-0 - disabling barriers
Setting up..done
```

► **Mounting file system**

```
Mounting local file systems...
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
tmpfs on /dev/shm type tmpfs (rw)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
kjournald starting. Commit interval 5 seconds
EXT3 FS on dasda1, internal journal
EXT3-fs: mounted filesystem with ordered data mode.
/dev/dasda1 on /boot type ext3 (rw,acl,user_xattr)
mount: /dev/dasdd1 already mounted or /opt busy
```

► **Networking setting for lo (loopback)**

```
Setting scheduling timeslices ..unused
Setting up hostname 'sles9x-tak'..done
Setting up loopback interface    lo
    lo        IP address: 127.0.0.1/8
..done
```

► **Starting several services**

```
Starting hardware scan on bootStarting CRON daemon..done
Starting Name Service Cache Daemon..done
..done
```

► **Boot process completed**

```
Welcome to SUSE LINUX Enterprise Server 9 (s390x) - Kernel
2.6.5-7.244-s390x (ttyS0).
itsolinux login:
```


10.3 Commands

The commands that can be used on the boot process are:

► **chkconfig**

Updates and queries runlevel information for system services.

The command **chkconfig** has five functions:

- Add new services for management.
- Remove services from management.
- List current startup information for services.
- Change the startup information for services.
- Check the startup state of a particular service.

► **dmesg**

Prints or controls the kernel ring buffer.

The command **dmesg** helps users to print out their bootup messages.

► **mkinitrd**

Creates initial ramdisk images for preloading modules.

The command **mkinitrd** automatically loads file system modules, IDE modules, and all `scsi_hostadapter` entries in `/etc/modprobe.conf`.

► **lsmod**

Shows the status of modules in the Linux kernel.

The command **lsmod** formats the contents of the `/proc/modules`, showing what kernel modules are currently loaded.

► **insmod**

Inserts a module into the Linux kernel.

The command **insmod** inserts a module into kernel.

► **rmmod**

Removes a module from the Linux kernel.

The command **rmmod** removes a module from the kernel.

► **modprobe**

Adds and removes modules from the Linux kernel.

The command **modprobe** looks in the module directory `/lib/modules/<kernel_version>` for all the modules and other files, except for `/etc/modprobe.conf` (configuration file) and `/etc/modprobe.d` directory.

► **zipl**

Boot loader tool for IBM S/390 and zSeries architectures.

zipl can be used to prepare devices for initial program load (IPL). It has the following supported functions:

- Booting a Linux kernel with optional ramdisk and kernel command line
- Taking a snapshot of the current system status (system dump)
- Loading a data file to initialize named saved segments (NSS)

The zipl tool implements a boot menu and includes the following features:

- Displays a list of available configurations
- Allows you to choose a configuration
- Allows you to specify additional kernel command-line parameters

► **fsck**

Checks and repairs a Linux file system.

fsck is used to check and optionally repair one or more Linux file systems. Normally, **fsck** will try to handle file systems on different physical disk drivers in parallel to reduce the total amount of time needed to check all of the file systems.

10.4 Boot problem examples

When we are talking about boot problem determination, there are basically two main problems:

► Linux cannot finish booting.

Processing stops somewhere in the boot process. There can be many reasons for this problem.

► The system does not work well after boot is completed.

The system has booted but Linux is not working properly, for example, the service that should start automatically does not start, some devices necessary for booting were not recognized, FCP devices cannot be recognized, LVM cannot be recognized, or the network is not connected.

For a better understanding and a definition of where the problem is, we use the diagram in Figure 10-7 to help to identify which phases on the boot process have problems.

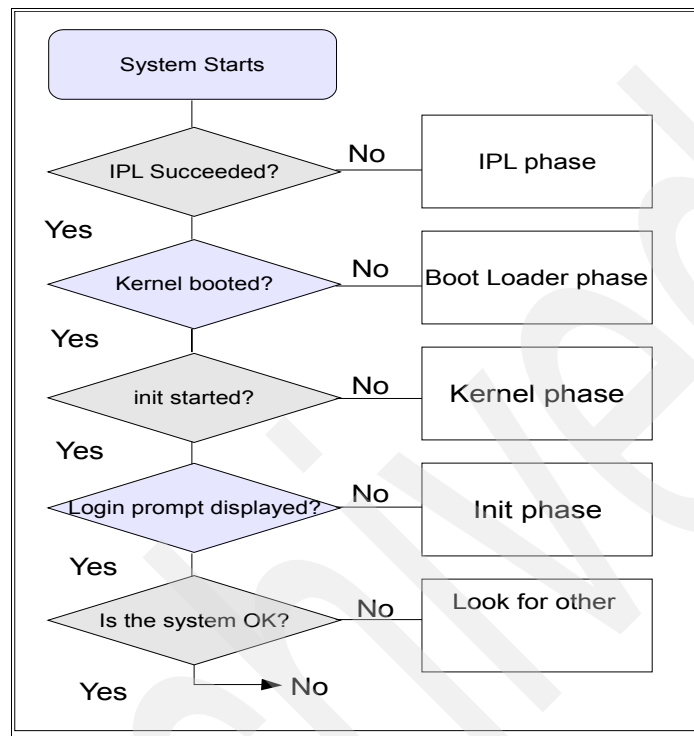


Figure 10-7 Diagram to identify boot problem

At this point, we have three possibilities: This occurs immediately after install, after changing the configuration, or after hardware failure.

Immediately after installation

The installation might not be correctly completed due to the trouble of the installer. After the installation ends, Linux does not boot it.

After changing the configuration

When the LVM configuration changes or disks are added, for example, necessary processing (execution of zipl and recreation of initrd and so on) is not executed.

10.5 Scenarios

This topic describes problems that can occur when you are booting your Linux system. For all examples, we use the same procedure to determine the solution.

First of all, we need to ask questions about what is causing the problem, and then we investigate.

10.5.1 IPL failure

Ask yourself the following questions when an IPL has failed:

- ▶ Problem description: IPL device not found
Questions to be answered:
 - Is the IPL address correct?
 - Is the IPL device address correct?
 - Is it defined on the IOCP of that LPAR?
 - Is the z/VM guest attached (under VM)?
- ▶ Problem description: Boot loader code does not exist in IPL device
Questions to be answered:
 - Is the IPL address correct?
 - Are the zIPL configurations correct?
 - Did the H/W trouble occur on the specified IPL device?

Possible solutions are:

- ▶ Confirm the connection to the IPL device.
- ▶ Re-IPL with the correct IPL address.

Execute the **zip1** command from the rescue system, and introduce the boot loader code again if the connection to the IPL device is good and this does not solve the problem.

Example 10-6 Message example in VM console - IPL with wrong address

```
Ready; T=0.01/0.01 23:06:59
IPL 3302 CLEAR
00: HCPGIR450W CP entered; disabled wait PSW 000A0000 0000000F
```

10.5.2 Failing in kernel booting

The following are thought to be causes of failure when the kernel is booting:

- ▶ Problem description: kernel image in boot loader code is destroyed.

Questions to be answered:

- Did the H/W trouble occur on the specified IPL device? Do hardware problem determination.
- Is the image file specified for the “image=” line in /etc/zipl.conf file normal?

- ▶ Problem description: kernel starts securing the memory more than the system defined.

Questions to be answered:

- Is the storage setting of LPAR or VM User correct?
- Did you define the “mem=” parameter in /etc/zipl.conf?

Possible solutions are:

- ▶ Reinstall boot loader code with the correct kernel image file by using the rescue system.
- ▶ Change the LPAR (or VM guest) storage size and re-IPL.
- ▶ With rescue booting, delete “mem=” in /etc/zipl.conf and re-run /sbin/zipl.

Example 10-7 Message example in VM console - kernel image is destroyed

```
z/VM V5.2.0    2007-01-16 13:12
Ready; T=0.01/0.01 23:37:21
IPL 3301 CLEAR
00: Booting default (ipl)...
00: HCPGIR453W CP entered; program interrupt loop
```

10.5.3 Failing since the init phase

The file systems are not able to mount (excluding root).

Init cannot skip a pertinent process, and stops during processing when there is a problem involving DISK (for example, when there is a partition problem).

- ▶ Problem description: failing in fsck

Questions to be answered:

- Is the /etc/fstab description correct?
- Are the devices correctly recognized?

- Does the file system break?
- Problem description: failing in mounting file system

Questions to be answered:

- Is the /etc/fstab description correct?
- Are devices correctly recognized?
- Does not the file system break?

Possible solutions are:

- Modify the fstab entries.
 - If there is a mis-setting, modify it.
 - If a problem in a file system occurs, comment it out temporarily or set the fsck flag to off (set it to 0).

- Re-make initrd.

The devices may not be recognized by coldplug but may be recognized after a re-make of initrd.

Example 10-8 Message example - failing in fsck

```
Checking file systems...
fsck 1.38 (30-Jun-2005)
Checking all file systems.
?/sbin/fsck.ext3 (1) -- /opt? fsck.ext3 -a /dev/dasdd2
fsck.ext3: No such device or address while trying to open /dev/dasdd2
Possibly non-existent or swap device?
fsck.ext3 /dev/dasdd2 failed (status 0x8). Run manually!
?1A..failedblogd: no message logging because /var file system is not
accessible
fsck failed for at least one filesystem (not /).
Please repair manually and reboot.
The root file system is already mounted read-write.
```

Attention: Only CONTROL-D will reboot the system in this
maintenancemode. shutdown or reboot will not work.
Give root password for login:

10.5.4 Failure to start a service

The following two cases are thought to be causes of failure when starting a service.

- ▶ Problem description: problem with automatic start setting.

Questions to be answered:

- Is the symbolic link file (whose name starts with Sxx) deployed in /etc/init.d/boot.d?
- Is the symbolic link linked with a correct start script?
- Is the symbolic link file (also known as symlink or soft link) deployed in /etc/init.d/rcx.d?
- Is the symbolic link linked with a correct start script?

- ▶ Problem description: The automatic setting is valid, but the service does not start.

Questions to be answered:

- Has the required service started?
- Is there problem in the start script?

A possible solution is doing an automatic start setting of the service. It is set so that the service starts when booting by using the **chkconfig** command or the **inserv** command.

Case study: slow responding Web site

In this chapter, we show how to determine the problem behind a slow response time and low transaction rate scenario in a typical three-tier Web-based serving environment, which is also a general problem usually reported by production administrators. Although we are restricted in our lab environment from recreating the problem, effort has been taken to reflect the real-world Web application production environment. The scope of this case study is to provide information about using various Linux and z/VM-based tools and other problem determination experiences in a combined manner.

11.1 Three-tier Web application environment

In this scenario we implemented a high-availability solution for a Web-based application with Linux images running on System z.

WebSphere Application Server (WAS) is one of the most common middleware products run on Linux on System z. WebSphere is also one of the most widely used Java-based application servers in the world, and many organizations use Web applications hosted on WAS to connect to data housed on the same mainframe.

To make the environment simpler, we used the Trade 6 application to recreate a Web Serving environment. The application data resides on DB2 for Linux on System z.

Important: We restrict this case study to only problem determination on the utilization of hardware resources. We also assume that the applications (IBM HTTP Server, WAS, and DB2) are optimally tuned to provide maximum performance.

11.1.1 Trade 6 Web serving application

The IBM Trade Performance Benchmark sample, also known as the Trade 6 application, is a sample WebSphere end-to-end benchmark and performance sample application. This Trade 6 benchmark has been designed and developed to cover WebSphere's programming model.

This application provides a real-world workload, driving WebSphere's implementation of J2EE™ 1.4 and Web services, including key WebSphere performance components and features. The Trade 6 application simulates a stock trading application that allows you to buy and sell stock, to check your portfolio, register as a new user, and so on.

In this book, we use Trade 6 to generate workloads that would impact the system performance. The IBM Trade Application for WebSphere Application Server is available to download for free at:

<https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=trade6>

The three main Trade 6 components are listed below and shown in Figure 11-1:

- ▶ IBM HTTP server

The HTTP server accepts client requests and delivers static content (HTML pages, images, and style sheets). Dynamic requests are forwarded to the WebSphere Application Server through a server plug-in.

- ▶ IBM WebSphere Application Server

The WebSphere Application Server creates dynamic content using JavaServer™ Pages (JSPs) and Java Servlets. Pages are generated from data extracted from a DB2 database. All Trade versions are WebSphere-version dependent, so Trade 6 only works with WebSphere Application Server V6.

- ▶ DB2 database

The DB2 database contains relational tables regarding simulated customer accounts and stock transactions. We used DB2 LUW v8.2.

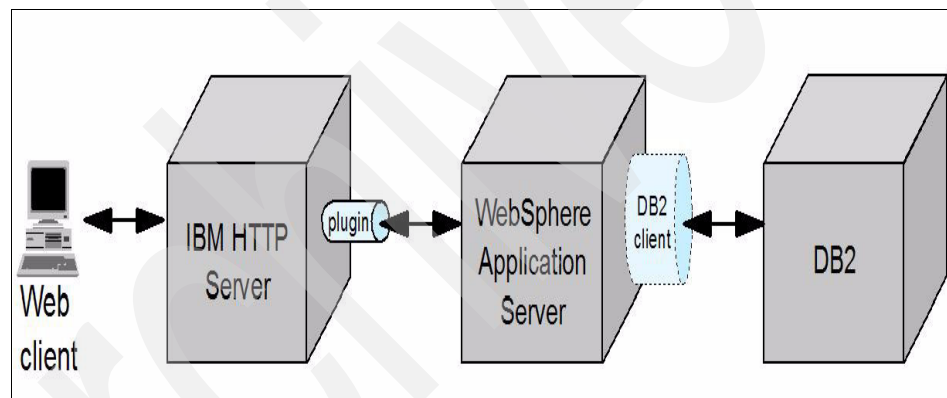


Figure 11-1 Components of a typical Trade 6 deployment

11.1.2 Our Web serving setup

Several options are available when deploying Trade 6 on Linux on System z:

- ▶ All components can run in a single Linux guest. This is referred to as a single-tier deployment.
- ▶ Each component can run in a dedicated Linux guest. We refer to this as a three-tier deployment.

We chose a three-tier deployment for this book, where we run the IBM HTTP server, the WebSphere Application Server, and DB2 in their own Linux guests (see Table 11-1 on page 317). Our deployment choice is depicted in Figure 11-2. Using a three-tier deployment enabled us to adjust the virtual machine size of each Linux guest more accurately, based on the task that particular guest performs.

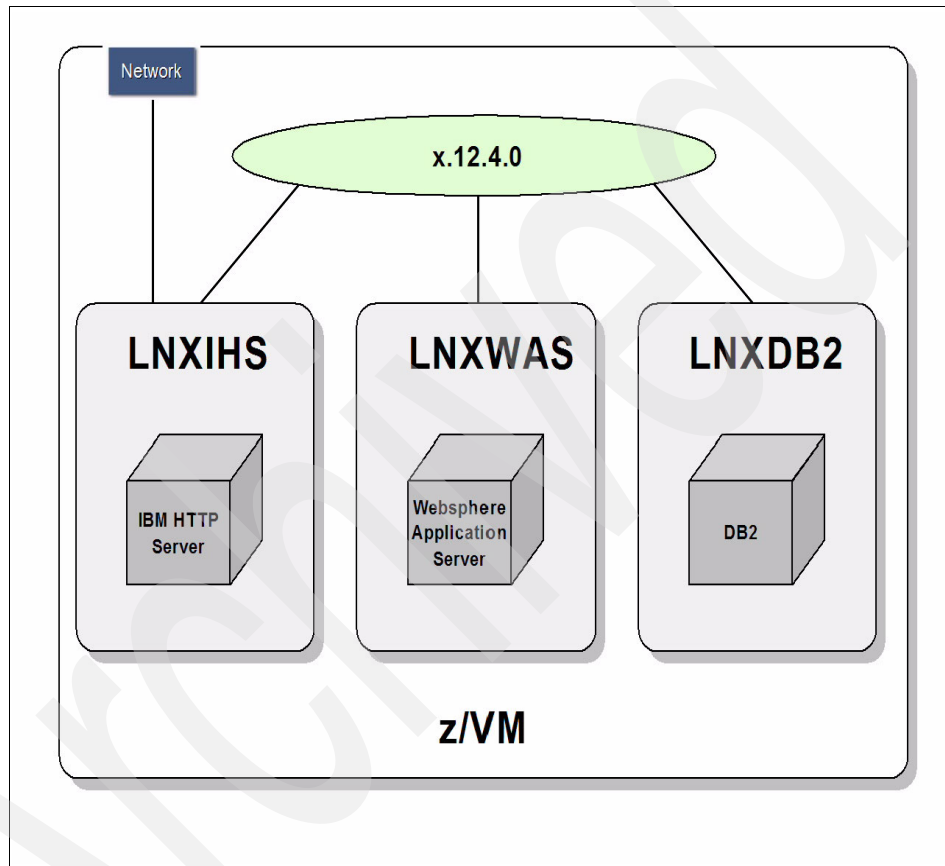


Figure 11-2 Three-tier deployment

In addition, when measuring utilization, the three-tier deployment option allows us to attribute specific resource usage to a specific task. See Table 11-1.

Table 11-1 *Linux servers and their configurations*

Server	Application	Processor	Memory
LNXIHS	HTTP Server	1	512 MB
LNXWAS	WebSphere Appl Server	1	3 GB
LNxDB2	DB2 Server	1	2 GB

11.1.3 WebSphere Studio Workload Simulator

The WebSphere Studio Workload Simulator for z/OS and OS/390® is an application that allows you to create multiple virtual or simulated users in order to test load and performance. The Workload Simulator consists of two components: a controller and an engine. For high scalability, Workload Simulator's Engine, which generates the load used during load-testing, is installed on a System z server. The load generated by the engine can be used to test any Web-serving environment (that is, the environment to be tested is not limited to z/OS). In our case, we tested against a WebSphere Application Server that was running on a Linux on System z guest. Workload Simulator supports multiple engines.

11.2 Symptoms of the problem

When we started to put load on the Web-serving environment, at one point of time the workload simulator reported that the HTTP server was not able to accept the page read request, and timed out. See a sample of the report that is generated in Example 11-1.

Example 11-1 *Report from workload simulator with timed-out messages*

```
04/01/2008    03:09:23
*****
04/01/2008    03:09:23    WebSphere Studio Workload Simulator
04/01/2008    03:09:23    Version 03353Z
04/01/2008    03:09:23    (c) Copyright IBM Corp. 2002,2003 All Rights Reserved.
*****
04/01/2008    03:09:23    IWL0075I Reading options from
/etc/iwl/common/default.conf.
04/01/2008    03:09:23    IWL0039I Script compilation SUCCESSFUL (180 ms)
04/01/2008    03:09:23    IWL0021I Engine name = Jibe
04/01/2008    03:09:23    IWL0030I Initial number of clients = 150
```

```

04/01/2008 03:09:23 IWL0080I Max. number of clients = 300
04/01/2008 03:09:23 IWL0032I Number of worker threads = 32
04/01/2008 03:09:23 IWL0040I Script file name = PingJDBCRead_Gold.jxs
04/01/2008 03:09:23 IWL0020I Percent element delay = 150%
04/01/2008 03:09:23 IWL0062I Startup delay = 1500 ms
04/01/2008 03:09:23 IWL0064I Max. connect timeout = 30 s
04/01/2008 03:09:23 IWL0065I Max. read/write timeout = 60 s
04/01/2008 03:09:23 IWL0068I Max. errors before exiting = 500
04/01/2008 03:09:23 IWL0077I Console print = on
04/01/2008 03:09:23 IWL0063I Validate length checking = on
04/01/2008 03:09:23 IWL0067I Dynamic DNS = off
04/01/2008 03:09:23 IWL0022I Dynamic cookies = on
04/01/2008 03:09:23 IWL0057I XML stats interval = 0 s
04/01/2008 03:09:23 IWL0041I Script repeat = 0
04/01/2008 03:10:25 IWL0046W HTTP read timeout encountered, server=9.12.4.243,
uri=/trade/servlet/PingJDBCRead, clientid=1.
04/01/2008 03:17:07 IWL0046W HTTP read timeout encountered, server=9.12.4.243,
uri=/trade/servlet/PingJDBCRead, clientid=18.

```

To diagnose this problem, we first tried a quick check with the **top** command on each server to see what was going on. The HTTP and DB2 servers appeared to have no problem in handling the workload. This can be seen by the way that the resources are being utilized in both servers, as shown in Example 11-2.

Example 11-2 HTTP server utilization

```

top - 18:01:05 up 3 days, 20:59, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 50 total, 2 running, 48 sleeping, 0 stopped, 0 zombie
Cpu(s): 1.0%us, 0.3%sy, 0.0%ni, 98.7%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 505168k total, 427472k used, 77696k free, 115524k buffers
Swap: 719896k total, 0k used, 719896k free, 129576k cached

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2598	nobody	20	0	231m	7016	1932	S	0.7	1.4	0:06.30	httpd
28770	nobody	19	0	232m	8152	1900	S	0.7	1.6	0:03.54	httpd
2600	nobody	21	0	230m	6836	1936	S	0.3	1.4	0:06.37	httpd
1	root	16	0	848	316	264	S	0.0	0.1	0:04.13	init
2	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.04	ksoftirqd/0

As we can see in Example 11-2 on page 318, the system is hardly utilizing the available processors, but, at the same time, it is able to manage the workload with the available memory efficiently. In the case of the DB2 server, however, both the processor and memory are not fully utilized, as shown in Example 11-3.

Example 11-3 DB2 server utilization

```
top - 18:15:30 up 3 min, 1 user, load average: 0.02, 0.03, 0.00
Tasks: 83 total, 2 running, 81 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.7%us, 0.3%sy, 0.0%ni, 99.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 2048816k total, 344008k used, 1704808k free, 7876k buffers
Swap: 719896k total, 0k used, 719896k free, 251156k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2390	db2inst1	15	0	172m	11m	8104	S	0	0.6	0:00.13	db2sysc
2413	db2inst1	15	0	172m	11m	8004	S	0	0.6	0:00.06	db2sysc
2462	db2inst1	15	0	172m	11m	7988	S	0	0.6	0:00.03	db2sysc
2469	root	16	0	2640	1300	980	R	0	0.1	0:00.05	top
1	root	16	0	848	316	264	S	0	0.0	0:00.50	init

Then we looked at the application server. It was consuming all of the memory that we had allocated to it (see Example 11-4). So we narrowed it down to the application server for further investigation. We also wanted to know the problem from the hardware resources perspective only.

Example 11-4 WAS server utilization

```
top - 18:15:30 up 3 min, 1 user, load average: 0.02, 0.03, 0.00
Tasks: 53 total, 1 running, 52 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.2%us, 1.3%sy, 0.0%ni, 0.0%id, 98.1%wa, 0.1%hi, 0.3%si, 0.0%st
Mem: 2048816k total, 1951484k used, 97332k free, 232k buffers
Swap: 719896k total, 719892k used, 4k free, 45300k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1521	root	17	0	2551m	1.5g	54m	S	21.7	78.5	2:24.74	java
68	root	15	0	0	0	0	D	1.7	0.0	0:00.40	kswapd0
1	root	17	0	848	68	44	D	0.3	0.0	0:00.51	init
4	root	10	-5	0	0	0	S	0.3	0.0	0:00.06	events/0
2	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
5	root	17	-5	0	0	0	S	0.0	0.0	0:00.00	khelper
6	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	kthread

8	root	10	-5	0	0	0	S	0.0	0.0	0:00.01	kblockd/0
26	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	cio
27	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	cio_notify
28	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	kslowcrw

11.2.1 Investigation of the problem

As discussed in the methodology, the first thing to do is to collect information about the system. So we run the **top** command to see the dynamic resource utilization. In Example 11-5, **top** reports that the CPU utilized by the WebSphere Application Server was around 1%. Memory allocated to the system is fully utilized. The swap field should be zero. If it is not, then the operating system is simulating the presence of more RAM by using disk space, which is a slow process compared to real storage.

Example 11-5 Resource utilization report of the WebSphere Application Server system using top

top - 21:47:10 up 29 min, 1 user, load average: 18.30, 6.12, 2.59											
Tasks: 54 total, 2 running, 52 sleeping, 0 stopped, 0 zombie											
Cpu(s): 0.3%us, 1.0%sy , 0.0%ni, 0.0%id, 98.3%wa, 0.0%hi, 0.3%si, 0.0%st											
Mem: 2048816k total, 2042076k used, 6740k free, 188k buffers											
Swap: 719896k total, 317948k used, 401948k free, 67316k cached											
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1500	root	17	0	2518m	1.6g	54m	S	0.1	81.4	4:47.58	java
1	root	17	0	848	80	56	D	0.0	0.0	0:00.52	init
2	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
4	root	10	-5	0	0	0	S	0.0	0.0	0:00.13	events/0
5	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	khelper
6	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	kthread
8	root	10	-5	0	0	0	S	0.0	0.0	0:00.01	kblockd/0
26	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	cio
27	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	cio_notify
28	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	kslowcrw
56	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	appldata
66	root	15	0	0	0	0	S	0.0	0.0	0:00.00	pdflush

To get more detailed information about the swap, we ran the **vmstat** command to record the overall resource utilization of the system, which included memory, swap, I/O, and CPU. The **vmstat** output in Example 11-6 shows a system in great pain. The **si** and **so** columns (swap-in and swap-out) are not even close to zero. This machine needs either more memory or fewer programs running. For more information about the **si** and **so** columns, run the command **man vmstat**. Notice also that the CPU is almost idle most of the time (on average, 97%).

Example 11-6 Resource utilization of application server using vmstat

procs		-----memory-----				---swap--		-----io-----		-system--		-----CPU-----				
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
0	0	713640	7784	1932	28644	0	0	0	7	152	142	0	0	99	0	0
0	0	713632	7784	1972	29096	1	0	1	5	158	141	0	0	99	0	0
0	0	713616	7536	2012	29040	1	0	1	8	159	143	0	0	99	0	0
0	0	713600	7536	2052	28984	1	0	1	3	161	140	0	0	99	0	0
0	0	713596	7412	2092	28916	1	0	1	8	156	143	0	0	99	0	0
0	0	713596	7412	2132	28876	0	0	0	2	152	140	0	0	99	0	0
0	0	719888	9144	2176	29316	1	210	1	217	257	142	0	0	98	2	0
0	0	718812	12976	2212	29316	122	26	122	32	186	174	1	0	97	1	0
0	0	718668	7340	2240	29228	360	164	360	172	270	259	1	1	93	5	0
1	0	719172	8764	2044	26808	789	453	789	456	360	339	2	2	86	10	0
0	0	718160	10364	808	21968	382	213	416	221	349	346	3	2	91	5	0
0	0	719400	10164	824	21124	264	200	264	202	425	366	3	1	92	4	0
0	0	719356	7364	864	21384	427	198	427	204	403	426	4	1	90	5	0
0	0	717484	7896	1276	27056	1503	603	1762	618	524	557	5	3	75	17	0
0	0	716004	8508	1008	26544	1365	673	1372	689	658	637	5	2	78	14	1
1	1	719880	8736	596	20972	2040	1103	2043	1106	834	2711	9	4	65	21	1
0	0	717044	12112	520	27192	2569	1200	2762	1210	779	897	6	4	59	30	1
0	0	719832	8780	496	21124	1820	982	1820	985	890	748	6	3	68	22	1
0	0	707740	7908	308	19468	2257	758	2317	765	777	808	6	4	69	21	1
0	0	719040	16928	460	22276	1867	1180	2015	1183	700	1280	6	3	65	25	1
0	0	706472	8556	312	18492	2620	860	2644	867	711	4182	10	4	64	20	1

From the above report we can see heavy swapping (a log of swap-in and swap-out activity), which is generally an indication of a problem with memory or disk. Usually, memory is mistakenly assumed to be the culprit in most situations because swapping involves writing to, and reading from, memory. However, it should always be taken into account what other component of the operating system is doing the work to make that activity possible, or maybe even necessary.

When we look at the situation in the context of the problem we are facing on the application server system, we have processors allocated that are under utilized even when there are heavy page requests from the application server. In our

case, the application server has already tried to consume more memory, which is also more than that of the actual real memory allocated, considering that the Linux kernel also is consuming some part of the real memory. That would also be a reason for heavy swapping activity as reported by the `vmstat` command.

11.2.2 Post investigation tuning

Based on our investigation, for the load we are putting on the application server, it seems to be consuming more memory. So we decided to increase the memory (over committing) for the VM guest to 3 GB. Then we started the workload with the same number of users that we opted for in the previous workload runs.

When the load reached the peak, we want to verify whether the problem is re-occurring. Also, we are monitoring the workload simulator for any of the errors that were generated earlier.

As shown in Example 11-7, we found that the VM guest where the application server was running was stable and there were no traces of swapping. The workload simulator was also generating the load without reporting any timed out error messages as seen earlier.

Example 11-7 System utilization with 3 GB memory, reported using the `vmstat` command

procs		-----memory-----				---swap---		-----io-----		-system--		-----cpu-----				
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
0	0	0	1713400	11108	236056	0	0	0	0	1079	516	6	2	92	0	0
0	0	0	1706332	11116	236048	0	0	0	7	734	531	5	2	92	1	0
0	0	0	1699512	11116	236048	0	0	0	0	504	518	5	2	93	0	0
0	0	0	1692444	11124	236040	0	0	0	12	522	531	5	2	92	0	0
0	0	0	1685624	11124	236040	0	0	0	23	892	535	5	1	92	0	1
0	0	0	1677192	11132	236032	0	0	0	7	506	507	8	2	90	0	0
0	0	0	1670000	11132	236032	0	0	0	28	543	549	6	2	92	0	0
0	0	0	1662808	11140	236024	0	0	0	11	536	542	5	1	92	0	1
0	0	0	1656732	11140	236024	0	0	0	0	553	568	6	2	92	0	0
0	0	0	1649416	11148	236016	0	0	0	7	533	541	5	2	92	0	1
0	0	0	1642100	11148	236016	0	0	0	0	543	551	6	2	92	0	0
0	0	0	1634280	11156	236008	0	0	0	7	560	565	7	2	90	0	1
0	0	0	1626336	11156	236008	0	0	0	0	563	584	6	2	91	0	0
0	0	0	1618648	11164	236000	0	0	0	36	551	576	6	2	91	1	0

11.2.3 What if the load is increased - does the problem reappear

To satisfy our own curiosity, we wanted to increase the load and check whether the problem reported was due to the lack of memory, which in turn starts the

swapping. So we increased the number of users or load on the environment. We also started to monitor and gather information about the servers.

After some time, when the load was at its peak, there were once again timed-out messages displayed on the Workload Simulator. But unlike the previous workload simulation, this time the **vmstat** tool did not report any swapping or high memory usage (see Example 11-8).

Example 11-8 vmstat report without any swapping with 3 GB memory

procs		-----memory-----				---swap---		-----io-----		-system--		-----cpu-----				
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
0	0	0	1561060	10516	235616	0	0	0	3	558	567	14	2	83	0	1
4	0	0	1476736	10548	235584	0	0	1	7	673	607	10	2	88	0	1
1	0	0	1379412	10572	235560	0	0	0	4	650	647	8	2	89	0	1
0	0	0	1273640	10600	235532	0	0	0	7	689	681	8	2	88	0	1
0	0	0	1175308	10616	235516	0	0	0	6	685	689	7	2	90	0	1
0	0	0	1078960	10636	235496	0	0	0	6	702	694	7	2	90	0	1
0	0	0	983356	10676	235456	0	0	0	3	680	684	11	2	86	0	1
1	1	0	886512	10704	235428	0	0	0	5	681	690	7	2	90	0	1
0	0	0	789720	10748	235900	0	0	0	5	683	696	7	2	90	0	1
0	0	0	693744	10788	235860	0	0	0	8	853	685	8	2	89	0	1
0	0	0	598140	10828	235820	0	0	0	2	723	688	7	2	90	0	1
0	0	0	500924	10872	235776	0	0	0	7	682	696	7	2	89	0	1
25	0	0	393160	10912	235736	0	0	0	2	672	681	12	2	85	0	1
0	0	0	392788	10952	235696	0	0	0	7	660	715	11	2	86	0	1
0	0	0	392788	10992	235656	0	0	0	3	658	694	7	2	90	0	1
0	0	0	392788	11032	235616	0	0	0	7	732	713	7	2	90	0	1
0	0	0	392788	11072	235576	0	0	0	4	678	710	9	2	88	0	1
1	0	0	392788	11112	236052	0	0	0	8	769	684	8	2	89	0	1
0	0	0	392788	11152	236012	0	0	0	3	672	703	7	2	90	0	1
0	0	0	392540	11192	235972	0	0	0	7	785	709	7	2	90	0	1

From Example 11-8, it appears that the system is stable and that there are no traces of swapping. As already recommended, we thought of cross-checking the utilization and other details from the z/VM level. Also, when things seem to be looking stable on Linux, but the problem is still occurring, it is better to gather information from the z/VM system for further investigation.

11.3 z/VM to the rescue

Even though we have many z/VM-based problem determination tools available, we directly started our investigation using the IBM z/VM Performance Toolkit,

since we wanted to compare and narrow down the problem. The IBM z/VM Performance Toolkit provides detailed system performance statistics in tailorable tabular and graphic formats. These reports are used for health checks, problem determination, and trend analysis. So during the workload simulator's peak load, for our further investigation we started to record and monitor the overall system using the IBM z/VM Performance Toolkit.

A good place to start looking at information in the Performance Toolkit is in the categories of general system data and I/O data. So the first thing we noticed was some indication of expanded storage paging. z/VM evolved around having a hierarchy of paging devices. Expanded storage is the high-speed paging device, and DASD the slower one where block paging is done.

This means that expanded storage can act as a buffer for more active users as they switch slightly between working sets. These more active users do not compete with users coming from a completely paged out scenario.

In this environment, real storage is sufficiently constrained so that there is heavy paging to expanded storage. However, expanded storage is sufficient so that DASD paging is essentially eliminated.

As we can see Figure 11-3, there is a drop in the I/O operation during the time period of our interest. With that as a base for further investigation, we compared the CPU load with the expanded storage paging rate.

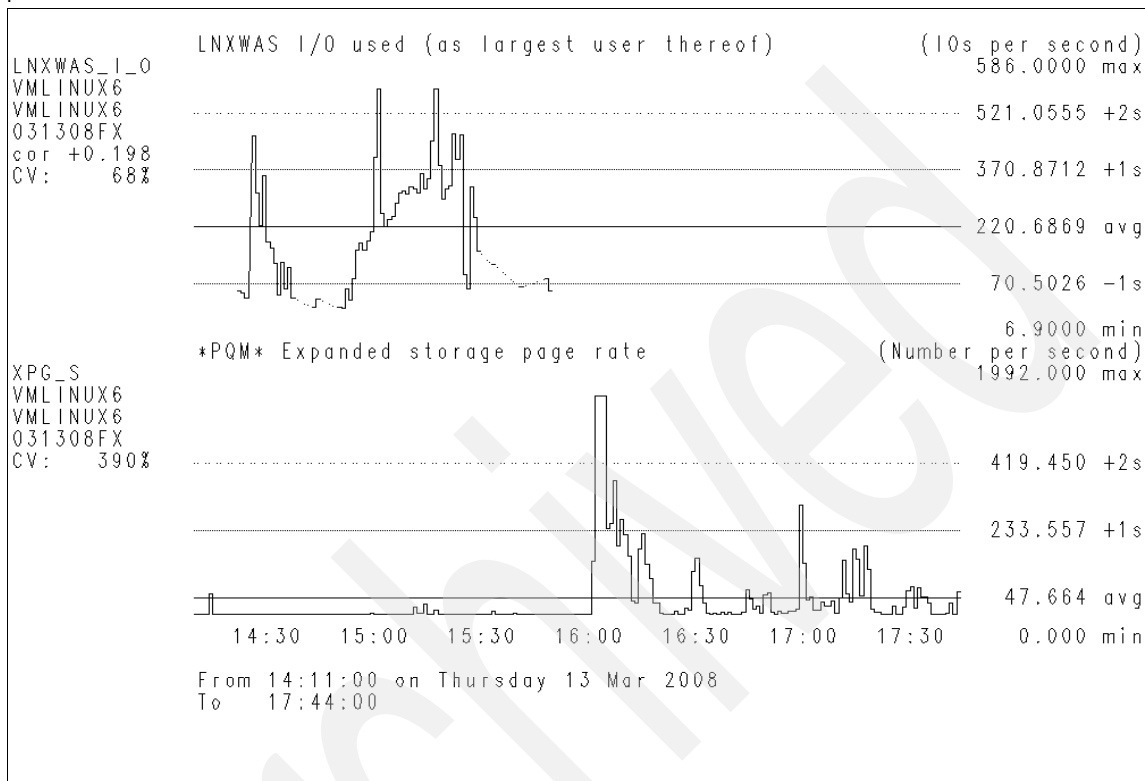


Figure 11-3 Example expanded storage page rate compared with I/O operation

Figure 11-4 shows a comparative chart of expanded storage with respect to the CPU load on the system. From the chart, it seems that at around 16:00 hours we have some expanded storage paging, and also at the same time we can see a uneven CPU load. So with this as an base, we started to compare various other resources during that particular time.

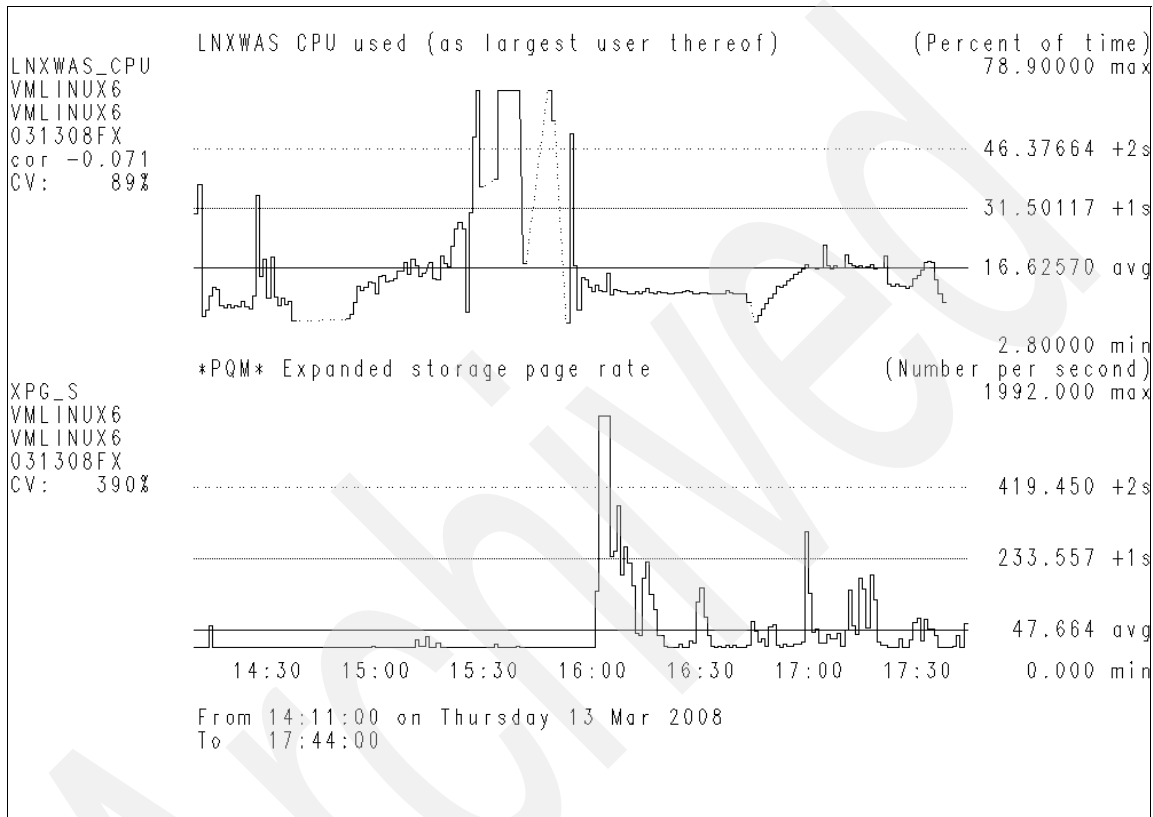


Figure 11-4 Example expanded storage page rate compared with CPU used

The next obvious statistical data of interest is the system page rate, as we can see in Figure 11-5. We can see that there is heavy system paging happening during the narrowed down timings.

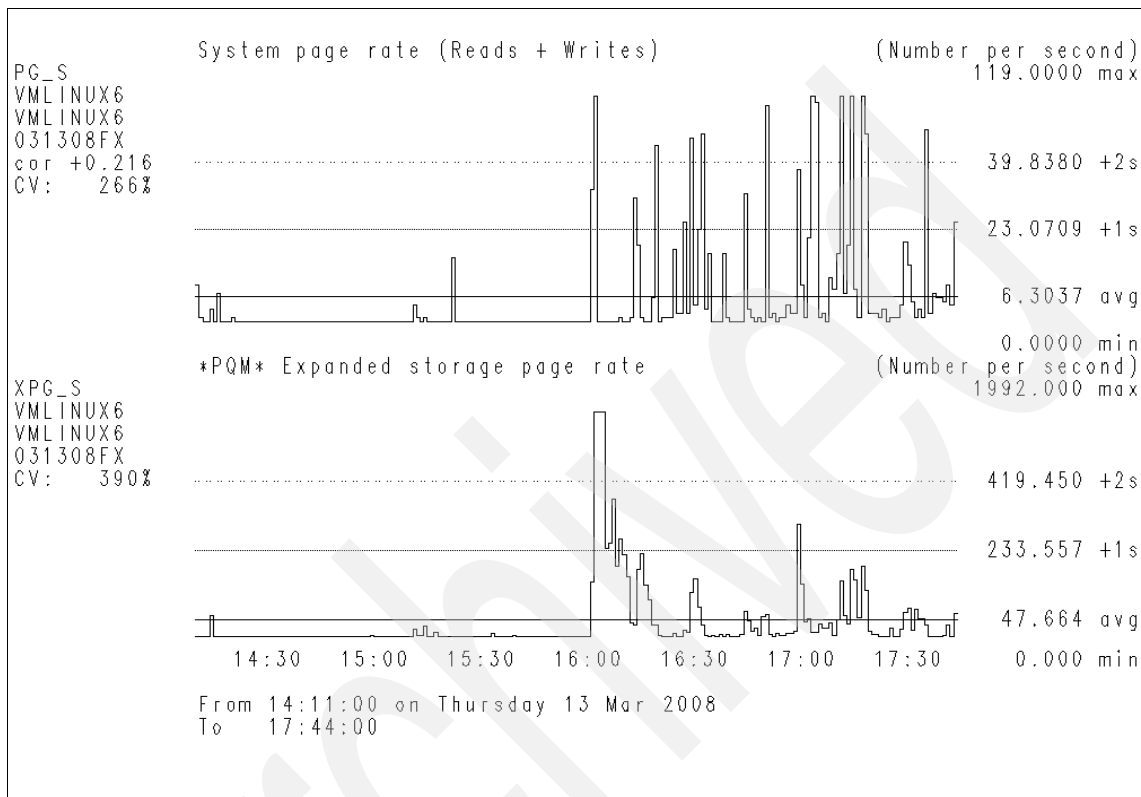


Figure 11-5 Example expanded storage page rate with system page rate

Increased paging can mean any of the following:

- ▶ The number of pages moved in/out of real storage per second (or per command) goes up.
- ▶ The number of page I/Os per second (or per command) goes up. Note that this is different from the previous point, since we can move multiple pages per single I/O.
- ▶ Paging to expanded storage stays the same or decreases, while paging to DASD increases.

Keep in mind that we page because we cannot fit everything in real storage at once. Therefore, paging increases are caused by:

- ▶ Decrease in available storage
- ▶ Increase in storage requirements
- ▶ A combination of both

QUERY FRAMES is the recommended way of checking storage, but it is only online. In our case, since the performance toolkit provides the same information as the paging load option, we started to monitor the paging activity from the performance toolkit.

Since we have high paging during the time period, we started to compare the system page rate with that of the other resources in use.

In Figure 11-6 we compared with pageable main storage in use. As we can see, the pageable main storage has jumped from the average utilization to the maximum utilization during the same time period. This is a usual phenomenon, since paging would start once the main storage gets filled.

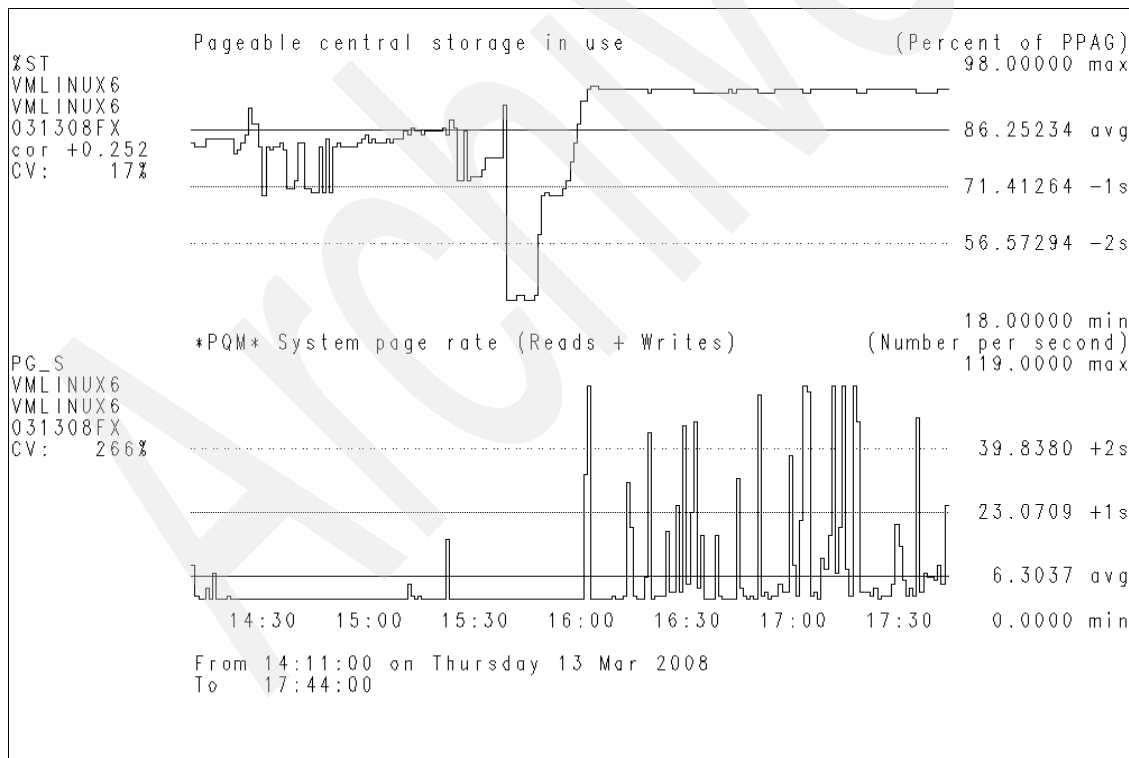


Figure 11-6 Example system page rate compared with pageable central storage in use

In Figure 11-7, we can see that when the storage page rate is started there is no or very limited virtual I/O operation happening. This is unusual since we are seeing that there is no I/O operation, which means that the processor is spending most of its time paging. This sharp variation in the resource utilization indicates some erratic and serious system behavior.

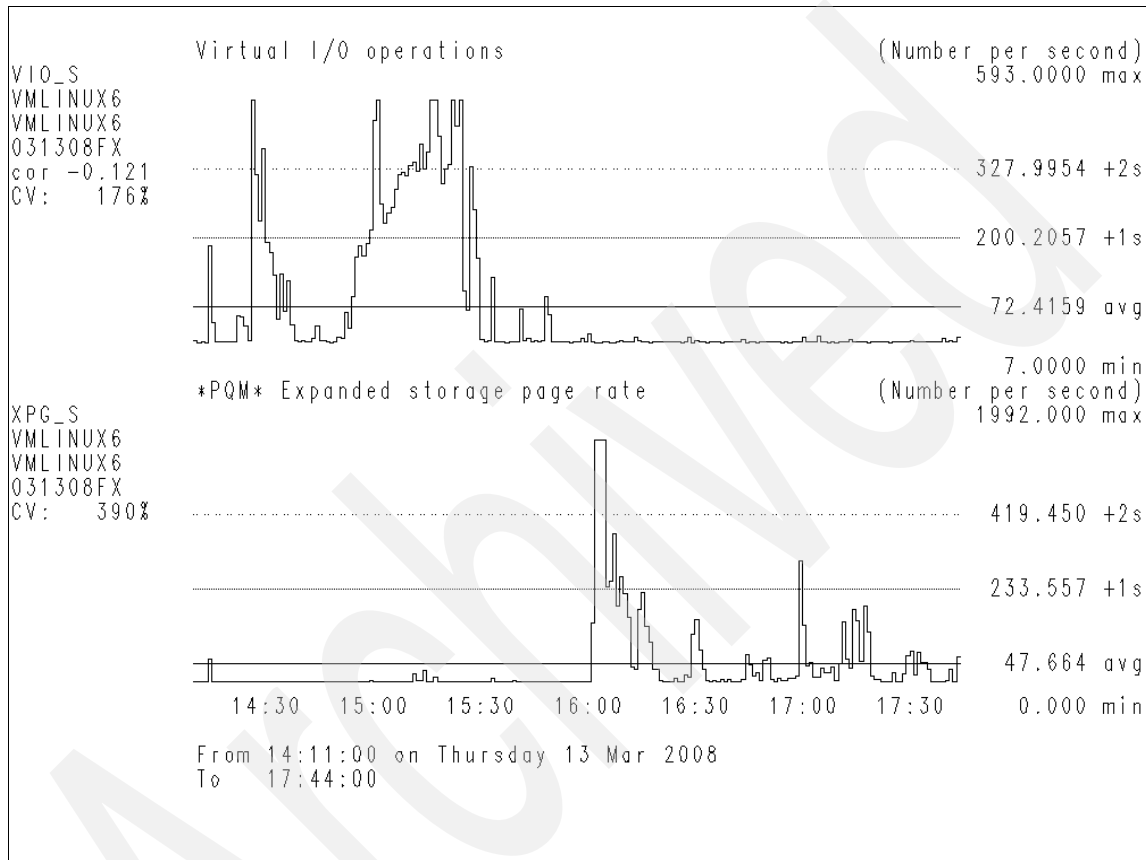


Figure 11-7 Example expanded page rate compared with I/O operations

11.4 Emergency scanning

When we drilled down further in the dynamic data provided by the performance toolkit, we witnessed some heavy emergency scanning happening in the z/VM. Emergency scanning is an indicator that the system is critically short of storage for some reason. When short of storage pages, the system begins a three-level scanning process: scan 1, scan 2, and then the most serious, emergency scanning. The system is being reduced to extreme measures while hunting for

main storage pages, and the situation is so severe that it will steal them from any user. Ultimately, whether they are inactive, in queue, or active, it eventually even steals pages from shared segments and the CP system process.

In short, the system is in a critical mess, storage wise. This also covers stealing pages from the monitor and MONDCSS, two critical performance-related entities, and in this event, performance monitoring and reporting may be compromised or interrupted.

Example 11-9 Example performance toolkit reporting emergency scanning (ESCN)

FCX101 CPU 2094 SER 2991E Interval 03 :10:00 - 03 :49:21 Perf. Monitor														
TIME >	PPAG	%ST	ALO/S	FPGS	%FR	SHAR	#TW	ESCN	%PGSL	%SPSL	XSTAV	%XS	XAL/S	XAG
15 :58	1000k	105	8	...	0	66k	0	100	14	66	1024M	99	15	476
15 :59	1000k	105	7	...	0	66k	0	100	14	66	1024M	99	22	568
16 :00	1000k	105	54	...	0	66k	0	92	14	66	1024M	99	163	191
16 :01	1000k	105	22	...	0	66k	0	100	14	66	1024M	99	68	200
16 :02	1000k	105	14	...	0	66k	0	100	14	66	1024M	99	38	267
16 :03	1000k	105	5	...	0	67k	0	100	14	66	1024M	99	25	367
16 :04	1000k	105	1	...	0	67k	0	..	14	66	1024M	99	0	654
16 :05	1000k	105	6	...	0	67k	0	..	14	66	1024M	99	0	12
16 :06	1000k	105	8	...	0	67k	0	100	14	66	1024M	99	22	928
16 :07	1000k	105	5	...	0	67k	0	100	14	66	1024M	99	14	11
16 :08	1000k	105	4	...	0	67k	0	100	14	66	1024M	99	9	13
16 :09	1000k	105	5	...	0	67k	0	100	14	66	1024M	99	10	14
16 :10	999k	105	5	...	0	67k	0	100	14	66	1024M	99	17	13
16 :11	1000k	105	6	...	0	67k	0	100	14	66	1024M	99	11	13
16 :12	1000k	105	5	...	0	67k	0	..	14	66	1024M	99	0	22
16 :13	1000k	105	1	...	0	67k	0	100	14	66	1024M	99	17	16

Usually an emergency scanning problem is detected when there are one or two guests (Linux servers) consuming the majority of the system pages available for allocation to the guest.

Also, some parts of VM 64-bit releases were still operating in 31-bit mode, which implies an upper addressability limit of 2 GB. This could cause certain runtime issues when running Linux guests, for example, when they were performing I/O. Linux has the unfortunate tendency to grab and hold on to all of the memory allocated to it, which sometimes leads to a large part of it being misused for internal I/O buffer and cache.

Linux guests, where a virtual storage equal to or greater than 2 GB is defined, are now common on many systems. In order to perform real I/O in this particular scenario prior to VM v5.2.0, the system needed to use areas below the requesting virtual machine 2 GB line, and, as these were already in use by the

virtual machine itself, the system had to resort to emergency scanning to find the storage pages that it required to complete the I/O task.

Figure 11-8 shows where there is heavy emergency scanning happening during the same time period as when we noted that there is some expanded storage paging.

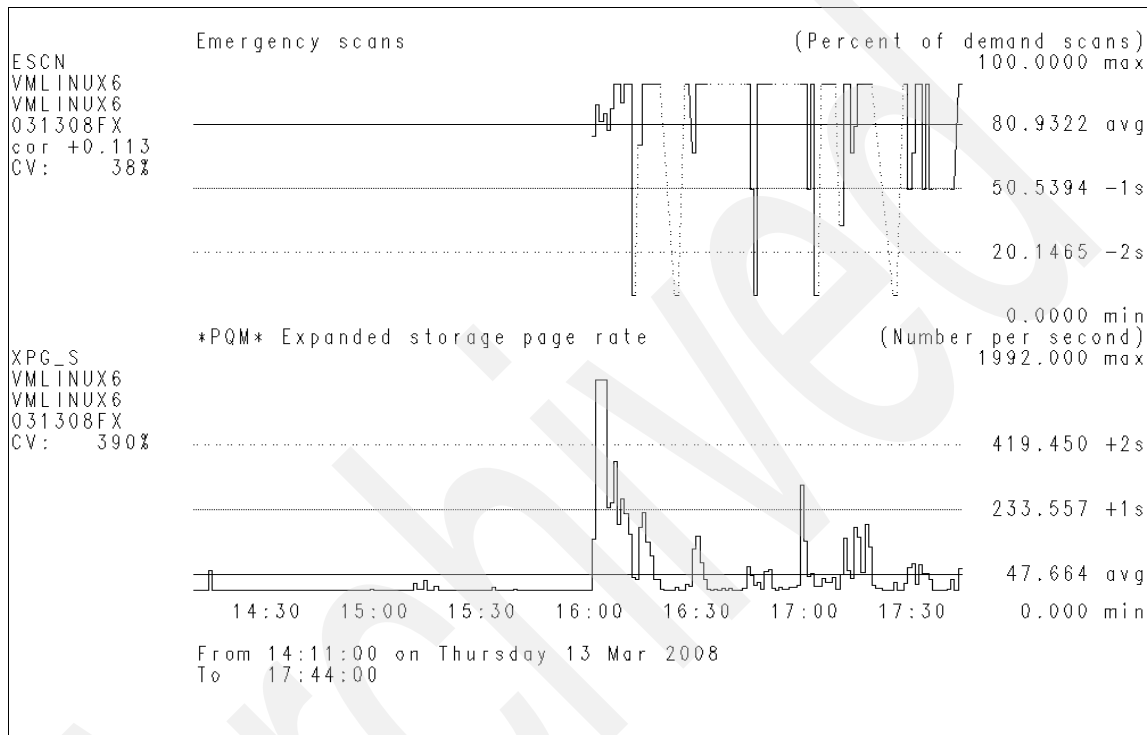


Figure 11-8 Example emergency scanning compared with exp storage page rate

Figure 11-9 shows an increase in the page rate. At the same time, we can see that z/VM has entered in to emergency scanning mode for release pages.

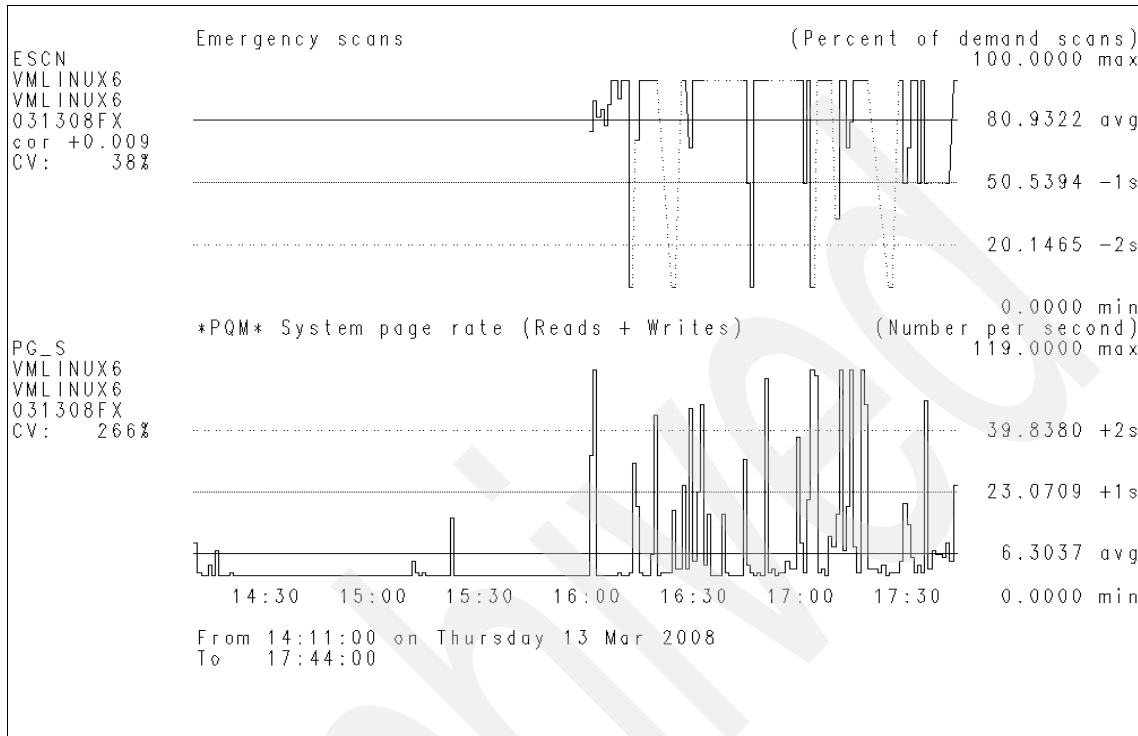


Figure 11-9 Example emergency scanning compared with system page rate

Note: Starting z/VM v5.3.0 introduced more important enhancements to CP storage management. Page management blocks can now be positioned above the 2 GB line. Contiguous frame management has been further refined, and fast available list searching was introduced in this release. For more discussion on this see:

<http://www.ibm.com/perf/reports/zvm/html/530stor.html>

From the above investigation, it is clear that the Linux guest (application server) is heavily committed in terms of memory within a 3 GB virtual machine, by defining a very large heap size for running the WebSphere Application Server. Some steps that you can take to generally tune your system and help avoid ESCN where it does still arise can be found at:

<http://www.vm.com/perf/tips/2gstorag.html>

Related publications

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

Online resources

These Web sites are also relevant as further information sources:

- ▶ For more information about IBM z/VM:
<http://www.vm.ibm.com>
- ▶ The dbginfo.sh script is available for download from IBM developerWorks:
<http://www.ibm.com/developerworks/linux/linux390/s390-tools-1.6.1.html>
- ▶ System status tool:
<http://pagesperso-orange.fr/sebastien.godard/>
- ▶ Linux man pages:
<http://linuxcommand.org>
- ▶ Installing Oprofile:
<http://oprofile.sourceforge.net/doc/install.html>
- ▶ GDB: The GNU Project Debugger:
<http://www.gnu.org/software/gdb/>
<http://sourceware.org/gdb/documentation/>
- ▶ For SLES10 information:
<http://www.novell.com>
- ▶ For RHEL5 information:
<http://www.redhat.com>

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Symbols

!!XGEN_DONT_EDIT_THIS! 109–117, 119, 121,
123–125, 127, 131–132, 135–137, 142–143, 146,
151, 153, 157–162

A

Application layer 92

B

barrier-based sync 302
blks 268, 299
boot process 267
boot.mult ipath 219
bottom line 41

C

cached 60
Case study 263
central storage (CS) 328
Channel Subsystem 72
command privilege (CP) 234
Common I/O
 Layer 245
complex problem 16
configuration file 270
 per-host basis 270
Consensus term 15
control program (CP) 22
Core Dump
 classic example 88
Core dump 87
CP command
 trace 28
CPU
 basis 180
 time
 180
 type 23
 utilization 53, 63, 170
cpu time 235

D

DASD
 device 80, 213, 292
 getting I/O statistical information 72
 statistics option 69
dasd driver 196
DASD dump 213
DASD Statistic 196
DASD volume 43, 83
 ordered list 43
dasdview command 214
DB2 database 224, 315
DB2 Server 317
Dec 18 19
 19
 04 xxxxxxxx kernel 257
default ERP 246
device 0.0.3600 264
Device D437 49
 Detailed Analysis 49
device driver 104, 244–245
device file 244, 247
Diag. X'98 47, 225
disk B 34
disk I/O
 bottleneck 192
 operation 192
disk storage (DS) 188
dispatch list 26, 234
dmesg command 297
dmesg print 298

E

elig 44, 224
Eligible List
 class users 235
eligible list 26, 234
EME 45, 225
end user 164, 239
Environment setup 89
error message 6, 256, 322
error recovery 199, 250
 action 200

- ESA 45
- etc/init.d/boot script 295
- EXPAN-001 E2-00000 24, 236
- EXPAN-001 E3-00000 24, 236
- Expanded storage
 - comparative chart 326
- expanded storage (ES) 23, 185, 324
- ext3 filesystem 180, 300
- EXT3 FS 302
- Extended IPv6 support 279
- Extended Link Service (ELS) 259

F

- FCP connection 193, 258
- FCP device
 - recognition 261
- FCP LUN 76
 - I/O statistics 80
- FCP port 261
- file system 8, 290, 292
- filesystem cache 189
- filesystems 306
- Firewall rule 279
- follow-up action 1, 17
- fsck command 292

G

- granular information 21, 59
- Graphical Data Display Manager (GDDM) 39

H

- H/W error 257
- H/W problem 260
- H/W trouble 308
- HCPGIR450W CP 35, 308
- Host Bus Adapter (HBA) 198
- HTTP server 314

I

- I/O act 47, 225
- I/O activity 67, 198
- I/O device
 - load 42, 196
 - performance 38
- I/O interruption 246
- I/O load 38
- I/O operation 170, 245, 325

- I/O rate 44, 224
- I/O request 247
- I/O time 70, 197
- IBM Tivoli OMEGAMON XE 52
- IBM z/VM
 - Performance Toolkit 21, 42, 323
- IBM z/VM Performance Toolkit
 - feature 38
 - initial screen 41
 - main menu 50
 - main screen 42
 - menu 45
- IBM z/VM Performance Toolkit commands
 - redispatch 50
- Ideal size 228
- Init phase 286, 292
 - little bit 293
- Initial Program Loader (IPL) 287
- initial set 43
- Input/output (I/O) 61, 181
- Inter process communication (IPC) 67
- IP address 280
- IP Forwarding 296
- IPL address 308
- IPL device 287
- IWL0046W HTTP 318

J

- JVM side 185

K

- kernel panic 35, 81
 - system crashes 81
- Kernel phase 290
- kernel space
 - callgraph feature 75
- kernel version
 - 2.6.16.21 76
 - 2.6.5 76
- ksoftirqd 180

L

- Linux 4, 21, 55, 59, 163, 211, 243, 270, 286, 313
- Linux commands
 - dasdview 213
 - lscss 212
 - lsdasd 213

- zipl 213
- Linux distribution 77, 270
- Linux environment 59
- Linux guest 33, 238, 315, 330
 - example SRM LDUBUF setup 239
 - paging volume 240
 - virtual machine size 316
- Linux image 60, 314
- Linux kernel
 - thread 180
- linux kernel 279, 322
- Linux side 187
- Linux system 31, 35, 66, 73, 171, 212, 282, 308
 - increases 180
 - move 171
 - running code 73
- Linux tools
 - callgraph 75
 - iostat 68
 - ipcs 67
 - Netstat 68
 - Opcontrol 74
 - OPProfile 73
 - ps 67
 - sysstat 62
 - top 66
 - tunedasd 72
 - vmstat 61
- linuxrc script 213, 292
- log level 102
 - zFCP driver 107
- Logical Volume
 - Manager 193
- ls command 96
- lscss command 212
- lsdasd command 213, 253
- lszfcf command 217
- ltrace utility 98
- LVM command 17
- LVM volume 292
 - group 301

M

- Machine type 260
- main stor 228
 - Ideal size 228
 - Max. size 228
- major/minor number 251

- Mar 10 12
 - 54
 - 05 linux kernel 279
 - 56
 - 32 linux SuSEfirewall2 279
 - 57
 - 40 linux SuSEfirewall2 280
- Mar 10 13
 - 00
 - 57 lxinst ifup 280
- MDISK cache 228
- Memory usage 52, 66, 231
- memory utilization 67, 170
- middleware 165
- min 66, 319
- MTU size 202

N

- named saved segments (NSS) 306
- Netfilter message 279
- NETIUCV driver 279
- network interface 68, 264
 - displays statistics 69
- networking option 170, 202
- ni 61, 324

O

- OMEGAMON display 52
- operating system 165, 286–287, 320–321
- Organizing the system (OS) 5, 92
- OSA Address Table (OAT) 140
- OSA card 203
- OSA device 250

P

- paging CHPIDs 240
 - one-to-one relationship 240
- paging operation 174
- Performance problem
 - CPU constraints 167
 - I/O resource constraint 167
 - memory 167
 - memory constraint 167
 - network utilization 170
 - operating system level 166
 - resource bottlenecks 164
 - software problems 164

- typical cause investigation flow 168
- performance problem 28, 57, 163–164
- Performance Problem Determination 163, 165
 - General flow 163
- Performance Toolkit 8, 21, 176, 323, 328
 - paging activity 328
- PF-key usage 41
- physical volume 267, 300–301
- problem determination 2, 9, 21, 59, 163, 165, 233, 269, 285, 313–314
 - analysis time 96
 - basic steps 10
 - first step 11
 - IBM z/VM Performance Toolkit 51
 - important aspect 14
 - meaningful data 43
- proc directory 104

Q

- query srm 27
- Queue Statistic 44, 224
- Queued Direct I/O (QDIO) 244

R

- Read-Copy Update (RCU) 291
- reader file 33, 36
- real storage 237, 320, 324
- Redbooks Web site 334
 - Contact us xiii
- Reserved pgs 47, 225
- resource utilization 57, 60, 166, 320
 - historic data 170
- response time 38, 62, 192

S

- S/W version 262
- same number 322
- Sample output 191
- sar 63
- SCSI device 215, 250, 301
- SCSI disk
 - dump tool 83
 - recognition 261
- SCSI trace 106
 - log level 108
- Secure Access Key (SAK) 101
- segmentation fault 93

- Service Level Agreement (SLA) 5
- Service time 49, 182, 196
- SET QUICKDSP 235
- Setup Problem 269
- single point of failure (SPOF) 3
- SLES9 SP3 217, 264
- software/program 287
- specified runlevel 293
- SPOF failure 4
- SPOOL pg 47, 225
- SRM STORBUF
 - 300 238
 - setting 240
 - value 238
- st 51, 62, 230, 244–245, 301, 318
- st 0 62
- st Mem 66, 318
- stand-alone dump 6, 83
- state matching 279
- storage page 44, 230, 329
- Storage problem 211
- storage server 195
- Storage Subsystem 72, 192, 195
- storage subsystem
 - practical limit 196
- SUSE Linux Enterprise Server
 - 10 Gallium 76
 - 9 SP3 76
- sy 78, 318
- symbolic link 261, 311
- system call 81–82
 - statistic 99
 - status 92
- system call status 83
- System z 4, 20, 59, 163, 172, 211, 243–244, 270, 286, 314
 - Linux installation 270

T

- t 24, 88, 182, 214, 295
- T=0.33/0.73 15
 - 24
 - 45 37
- TEACHOR1 QDIO
 - 0601 OSA 47
 - 0650 OSA 47
- three-tier deployment 315
- trace information 105

trans. time 44, 226
turn around time (TAT) 3

U

user LNXDB2 225
 Detailed data 225
User resource usage
 page 45
User Status 44, 224
userid LXOR4 46
 detailed status 48
Utilities Reference 27

V

var/log directory 277
var/log/boot.msg file 297
VF emulation 47, 225
VF overhead 47, 225
VG 222
virtual machine
 CP class 23
 performance monitors 40
virtual machine (VM) 23, 234–235, 316
VM guest 170, 308, 322
VM user 309
vmstat output 176–177, 321
volume table of contents (VTOC) 213
vswitch 23

W

Working Set Size
 Large values 188
Working Set Size (WSS) 188
Workload Simulator 317

X

XAL/S (XS) 51, 230, 330
XSTORE page 225

Z

z/VM system 23, 55, 138, 231, 323
 overall status 23
 real time performance analysis 40
z/VM system tools
 cp close console 33
 cp query 31
 dumpload 36

indicate 23
indicate load 23
indicate paging 24
indicate paging wait 25
indicate queues 26
paging wait 24
q trace 30
query alloc page 28
query srm 27
trace 29
vmdump 33
zFCP
 steps to gather statistical information 77
zFCP Statistic 76–77, 198
zFCP trace 106
 log level 107



Problem Determination for Linux on System z



Redbooks®

Problem Determination for Linux on System z

**Learn a Linux on
System z problem
determination
methodology**

**Discover problem
determination tools
for z/VM and Linux
on System z**

**Do a case study
using what you learn
here**

This IBM Redbooks publication addresses some of the common problems that customers have experienced on the Linux® on System z™ platform. This book provides a problem determination methodology and tools to help the reader diagnose the problem in an easy-to-read self-help manual.

We start with a discussion on how to approach problem solving in the Linux on System z environment and continue on to describe some of the problem determination tools commonly used for z/VM and Linux on system z. We continue with discussions on network problem determination, performance problem determination, and storage problems.

Additionally, we discuss the formation of eligible (or eligibility) lists.

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