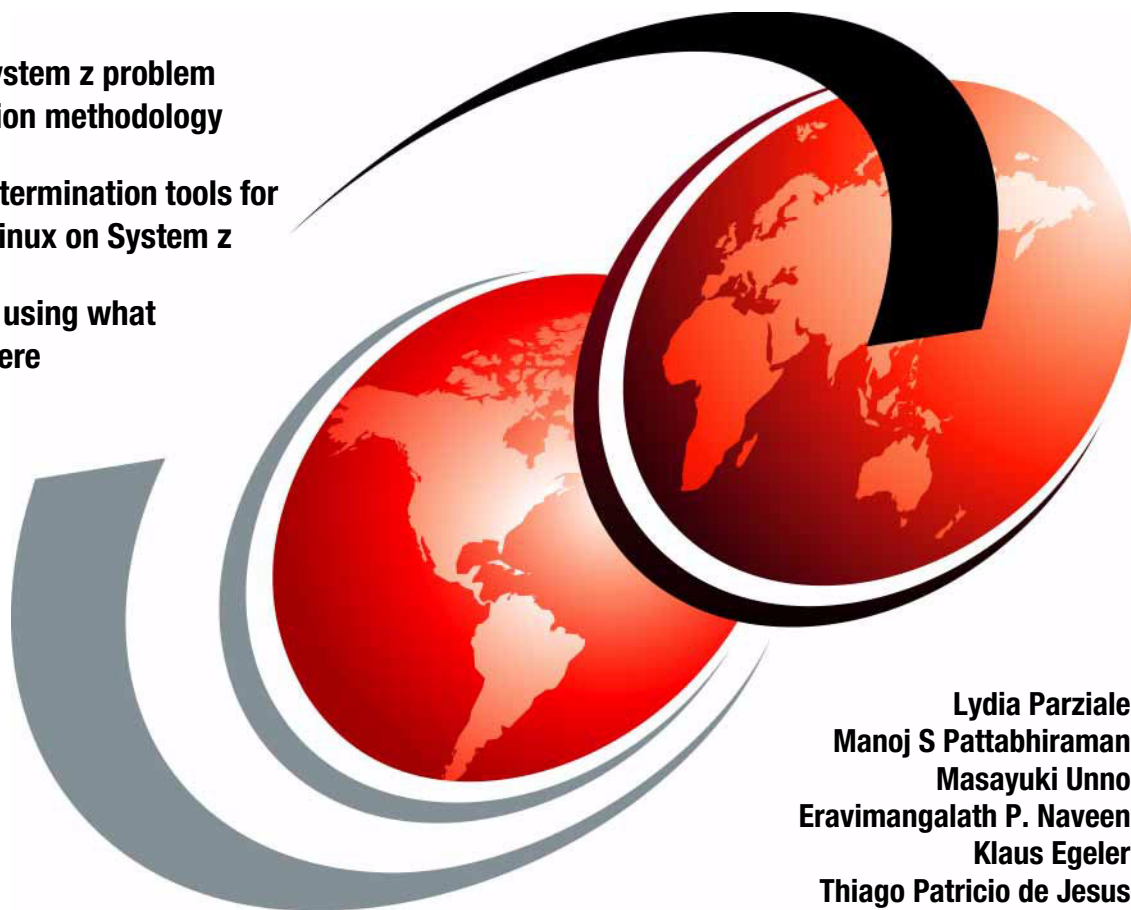


Problem Determination for Linux on System z

Linux on System z problem
determination methodology

Problem determination tools for
z/VM and Linux on System z

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

March 2008

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

First Edition (March 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.

This document created or updated on June 13, 2008.

© 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	1
This book The team that wrote this book	1
Become a published author	2
Comments welcome	3
 Chapter 1. Problem determination methodology	1
1.1 Proactive approach for minimization the effects of problems	2
1.1.1 Design and implementation phases	2
1.1.2 Test Phase	6
1.2 Problem determination basics	8
1.2.1 Basic problem determination steps	9
1.2.2 Gather information /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 support team to analyze	16
1.2.6 How to approach to complex problems	17
1.3 Problem resolution and follow-up actions	18
1.3.1 Expanded check for other systems	18
1.3.2 Update	19
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	27
2.1.5 VM trace	28
2.1.6 vmdump	32
2.1.7 IBM z/VM Performance Toolkit	37
2.1.8 IBM Tivoli OMEGAMON XE on z/VM and Linux	50
 Chapter 3. Problem determination tools for Linux on System z	57
3.1 Information gathering tools	58
3.1.1 dbginfo.sh	58
3.1.2 vmstat	59

3.1.3	System Status (sysstat) tool	60
3.1.4	top	64
3.1.5	ps	64
3.1.6	ipcs	65
3.1.7	iostat	65
3.1.8	netstat	66
3.1.9	DASD Statistics	67
3.1.10	OProfile	70
3.1.11	zFCP Statistics	73
3.1.12	Summary of information gathering tools	77
3.2	Dump and Trace	78
3.2.1	When to take Dump?	78
3.2.2	When should a Trace be run?	79
3.2.3	Standalone Dump	80
3.2.4	Verification of the dump	82
3.2.5	Core Dump	84
3.2.6	Summary of Dump techniques	89
3.3	Debugging Tools	89
3.3.1	gdb	90
3.3.2	strace	93
3.3.3	ltrace	95
3.3.4	SysRq	97
3.3.5	s390dbf	101
3.3.6	zFCP/SCSI	103
Chapter 4. Network <u>problem determination</u>Problem Determination . . .		107
4.1	<u>networking option:Overview</u> Overview of networking options for Linux on System z	108
4.1.1	Open Systems Adapters	108
4.1.2	HiperSockets	110
4.1.3	Guest LAN	111
4.1.4	Virtual Switch	112
4.1.5	<u>networking option:Summary</u> Summary of networking options for Linux on System z	113
4.2	Network interface detection and configuration under Linux on System z	113
4.2.1	The Linux hotplug system	114
4.2.2	Flow of network interface detection and activation under Linux . . .	114
4.2.3	Network interfaces and their interface names	118
4.2.4	Network configuration files overview	120
4.3	The art of network problem determination	121
4.3.1	Problems related to network communication	122

4.4 <u>Case study</u> Case studies	154
Chapter 5. Performance Problem Determination	157
5.1 What is a performance problem?	158
5.2 General flow of performance problem determination	158
5.2.1 Data necessary for investigation	163
5.2.2 CPU bottlenecks	164
5.2.3 Memory bottlenecks	174
5.2.4 I/O bottlenecks	183
5.2.5 Case studies	196
Chapter 6. Storage problems	201
6.1 How to determine a storage related problem	202
6.1.1 How to check DASD devices	203
6.1.2 How to check FCP/SCSI devices	206
6.1.3 What to check for multipathing	208
6.1.4 What to check for LVM	210
6.1.5 I/O statistics	212
6.2 Case study	213
Chapter 7. Eligibility Lists	227
7.1 CP Scheduler	228
7.1.1 Thrashing	230
7.2 SRM CONTROLS	231
7.2.1 SRM STORBUF	231
7.2.2 SRM LDUBUF	232
7.2.3 QUICKDSP	233
7.3 Other Recommendations	234
7.3.1 Example scenario	235
Chapter 8. Hardware related problems	237
8.1 Basic Configuration of Hardware.	238
8.2 Device recognition and management	239
8.2.1 Whole image of device control	239
8.2.2 Kernel Layer	240
8.2.3 Device Driver.	243
8.2.4 Common I/O	244
8.2.5 QDIO	248
8.2.6 Confirming Status of Channel-path and Devices	248
8.3 Hardware Problem Determination	251
8.3.1 Initial determination of hardware or software problem	251
8.3.2 Investigation into the H/W related messages	253
8.3.3 Approach for H/W problem	256
8.3.4 Actions after dintinction with S/W or H/W problem	259

8.4 Case Study	260
8.4.1 Memory card trouble	260
8.4.2 OSA trouble	260
8.4.3 DASD error	262
8.4.4 Network interface error in reboot	263
8.4.5 DASD recognition trouble in boot process	264
Chapter 9. Installation and Setup Problems	267
9.1 Understanding the installation process	268
9.2 Gathering information regarding installation process	273
9.3 Scenarios	281
9.3.1 Trouble during the installation	281
9.3.2 Trouble after the installation	282
Chapter 10. Booting Problems	283
10.1 Understanding the boot process	284
10.1.1 IPL phase	284
10.1.2 Boot Loader phase	285
10.1.3 Kernel phase	288
10.1.4 Init phase	291
10.2 Where gather information regarding boot process	294
10.3 Commands	303
10.4 Boot Problem examples	305
10.5 Scenarios	307
10.5.1 IPL failure	307
10.5.2 Failing in Kernel Booting	308
10.5.3 Failing since the init phase	308
10.5.4 Failing in starting a service	310
Chapter 11. Case Study : Slow responding website	311
11.1 Three-tier Web application environment	312
11.1.1 Trade 6 Web serving application	312
11.1.2 Our Web serving setup	313
11.1.3 WebSphere Studio Workload Simulator	315
11.2 Symptoms of the problem	315
11.2.1 Investigation of the problem	317
11.2.2 Post investigation tuning	319
11.2.3 What if the load is increased - does the problem re-appear?	320

11.3 z/VM to the rescue 321

11.4 Emergency Scanning 327

Related publications 331

IBM Redbooks 331

Other publications 331

Online resources 331

How to get Redbooks 332

Help from IBM 332

Index 333

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Trademarks

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

Redbooks (logo) ®
AIX®
DB2®
developerWorks®
ECKD™
ESCON®
eServer™
FICON®
GDDM®

HiperSockets™
IBM®
Lotus®
OMEGAMON®
OS/390®
Redbooks®
REXX™
S/390®
System z™

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

The following terms are trademarks of other companies:

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.

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 some 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 Masters 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 AP 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, 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- 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 7 years of experience in Software Group field. He holds a degree in Electrical Engineer with specialization in Eletronic and a Technology and Information Systems Latu Sensus 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, Naoshi Kubo
System z ATS, IBM Japan

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.

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks 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



1

Problem determination methodology

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

Figure 1-1 on page 2 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.

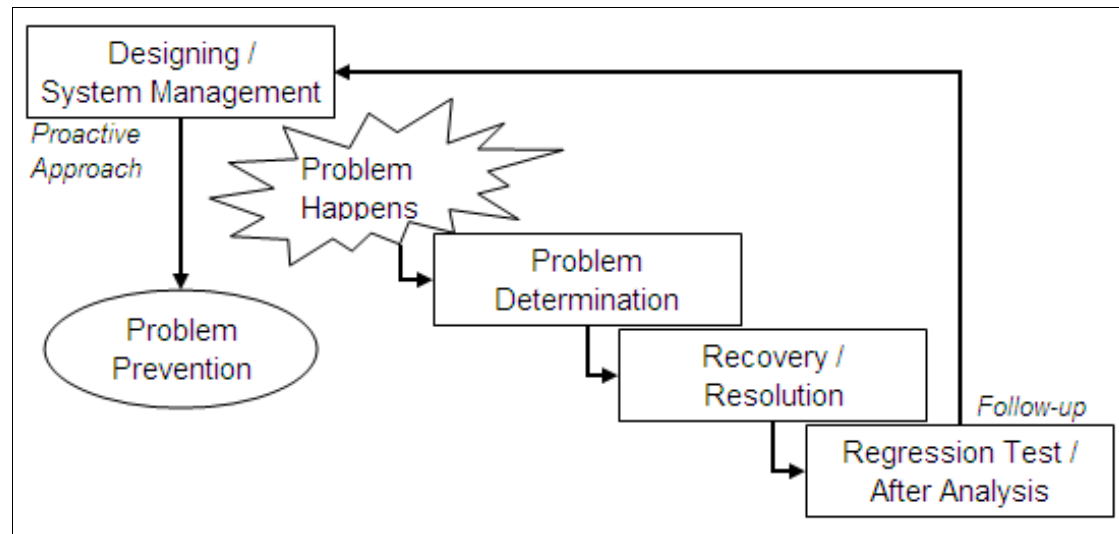


Figure 1-1 Life cycle of problem determination

1.1 Proactive approach for minimization 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 etc.

Designing, system management, and preparation for problems would minimize 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 phase.

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

- Installation and settings for emergency information collection

Please 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, if you can. 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.

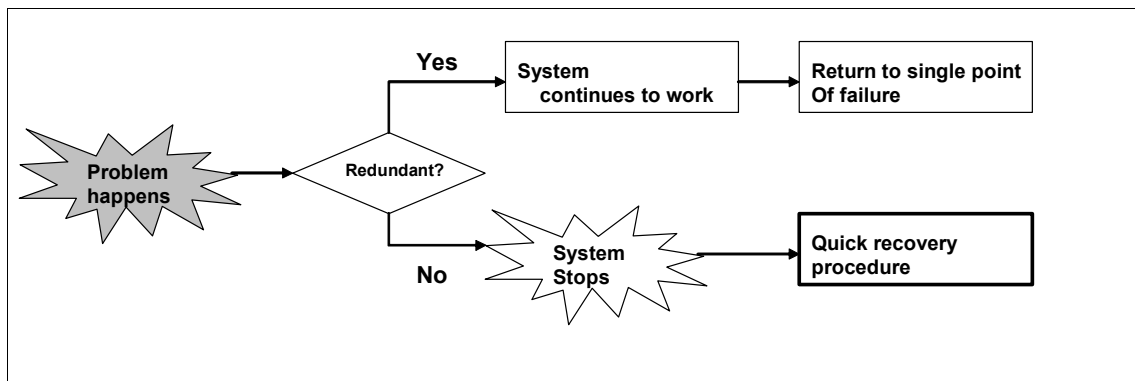


Figure 1-2 Necessity of quick recovery procedure.

A rescue Linux image will help in recovering 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.

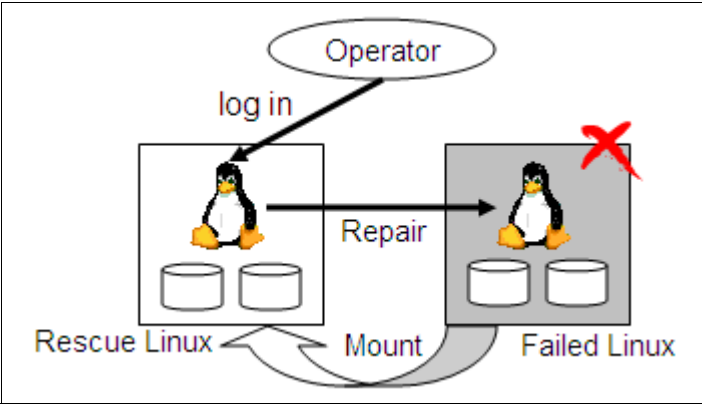


Figure 1-3 Sample image of recovery with rescue Linux

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 is a good hint to find the cause of service level degradation.

This table is a sample template of system management monitoring requirements. Please 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 help analysis of the problem.
	A-2) Automated Keeping		Storing the corrected data automatically in specified term. Removed the expired data automatically
	A-3) Data Processing		For performance data, it should be easy to process for analyzing, for example, organized on Excel® sheet.

B) System Monitoring	B-1) Monitoring Contents	B-1-1) Performance Monitoring	Monitoring the performance status based on threshold. (e.g. Overconsumption system resource.
		B-1-2) Service Monitoring	Monitoring the status of service and system. Alert to operator if these stop or get in bad condition.
		B-1-3) State Transition Monitoring	Notice to operator about status transition. (e.g. Takeover on redundant system, Batch Job ends successfully
	B-2) User Interface		Operator can monitor easily with graphs or charts etc.
	B-3) Service duration		Monitoring 24 by 365

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 which 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 packages 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 etc.)
- ▶ 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 will 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
- ▶ Practice gathering informations

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 the 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 failure test:

- ▶ System behavior
 - System behavior is same as expected.
 - Extent of impact is same as expected.
- ▶ Resource usage
 - Resource usage is same as expected.
 - Resource usage increase has side effects.
 - Performance decline of own application.

- Performance decline of the other systems which share same resource.
- Causes a critical issue (ex: cpu overload, I/O hang, thrashing etc.)
- ▶ Redundancy check
 - System continues or recovers as expected.
- ▶ Error code and messages
 - Error code and messages are same as expected.
 - Choose messages which should be monitored.
 - Critical message is described in messages and code manuals.
- ▶ Recovery procedure
 - Recovery procedure works correctly.

Table 1-3 represents some of the potential problems that can be practiced 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 do it. You may want to perform these rehearsals during the failure tests mentioned previously.

We suggest confirming the following during your rehearsals:

- ▶ Operation

- You should practice using the IBM z/VM Performance Toolkit (see 2.1.7, “IBM z/VM Performance Toolkit” on page 37) and the Linux sysstat package (see 3.1.3, “System Status (sysstat) tool” on page 60) . For example, you could create some z/VM raw monitor data and read it into the IBM z/VM Performance Toolkit .
- ▶ Data gathered is correct
- ▶ Additional system load (your system should have enough of each of the following in order to collect the data you need)
 - CPU and Memory
 - File size (enough for file systems)
 - Duration time
- ▶ Contact point and formations in an emergency
- ▶ Organizing the system information
 - Version list of OS, Microcode, Middleware, Packages, PTF etc.
 - 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 is the problem” and “Why the problem happened”.

- ▶ “What is the problem”
 - Understand what the problem is, and obtain a consensus with the team, both IBM and customers.
 - Gathers correct information related to failure component(s).
- ▶ “Why the problem happened”
 - Specify the cause or trigger of the problem with supporting information (logs, traces etc.).
 - 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 right 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 to it. 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 of analysis, as shown in this figure. “Report” is not only reporting the result but also includes discussion about follow-up actions.

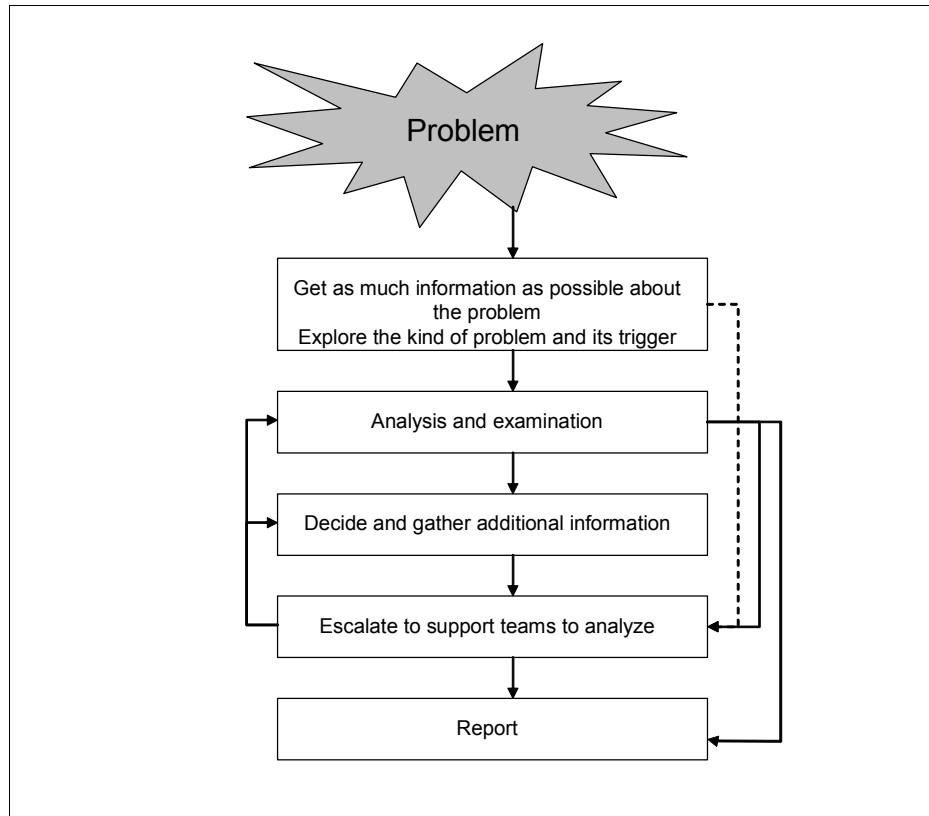


Figure 1-4 Basic steps of problem determination

You would determine which component is failing in the order shown in Figure 1-5, where you would first look at the Application or Middleware and finally look at the actual hardware to determine where the failure is originating.

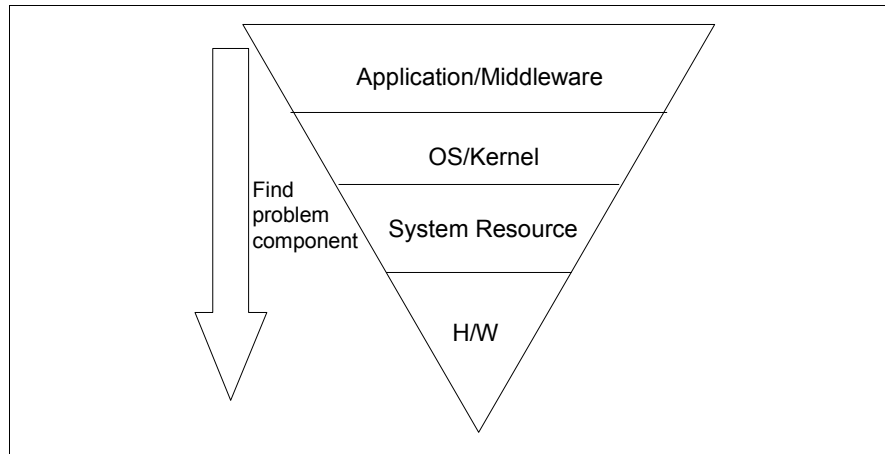


Figure 1-5 Find the problem component

1.2.2 Gather information /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 on what data to collect. This is the first step of problem determination. Often the cause of the problem can be found at 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

You will need to collect as much information as possible about the problem. Some of this information would be::

- ▶ A complete description of the problem
- ▶ Date and time of the problem
- ▶ Which system did the problem appear on? Does it appear 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)
 - Command or script log should be referenced.
- ▶ Current status (ongoing problem / recovered / avoided)
 - Whether there is a recovery plan or plan for avoidance of the problem.
- ▶ Scope (multi-system or not / multi-function or not)
 - Common terms or differences
- ▶ Difference from daily operation (loading, operation order etc.)
- ▶ Past results
 - Target operation has had a similar result or is this a new behavior.
 - There are any change from last results, or exactly same.
- ▶ Same/similar problem
 - Same or similar problem happened before?
 - If it has happened before, check necessary information, avoidance method, and recovery procedure.
 - If it happens and is already 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/ all failure)

Information that should be gathered

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

- ▶ Define the symptoms of the problem
 - ie: boot failure, disconnection network, I/O failure, etc.
- ▶ Find error component
 - Hardware, kernel, application, etc.

- We may have a chance to find the error component quickly based on the symptoms, error messages, or other signs.
- ▶ Confirm 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 etc.
- ▶ 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:

1. Determine the cause of the problem
2. Determine a method to reproduce the problem
3. Determine an avoidance method (or workaround) 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 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 if you may 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 understand the following things:

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, or Network
 - Check other transactions or processes running in parallel.
3. Prepare the test system
 - Use the test system
 - Use 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 Chapter 1.2.6, “How to approach to complex problems” on page 17

Determine an avoidance method

An avoidance method is a temporary solution or work around for a problem. For example, temporarily changing the operation, 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 will 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 workaround 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 agreement or 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 things before you can collect additional information:

- ▶ Was anything found regarding the problem using 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 the 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 reproduction test to see if the problem recurs.
 - Set tools for gathering informations in 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 system
 - Additional system load of test and gathering 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 etc.)
 - Effects on operation
 - Change scope of operation, configuration, script
 - Problem detection method when problem reproduction

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

1.2.5 Escalate to 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. They will ask you to provide additional information. The information that they may need is:

- ▶ Frequency
 - Date when the problem happened
 - How often did the problem happen
- ▶ 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 to complex problems

Problem reproduction helps in determining the exact cause of the problem. If the problem isn't reproducible, it is difficult to find the cause. It's important to gather the correct and enough 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 doesn't reproduce the problem for a specified amount of time, change to another type of test case.

Sample test case:

- ▶ LVM command hangs
 - Prepare a Linux test system which has the same version of kernel, LVM, and DM.
 - Create a script that runs the LVM commands.
 - It doesn't include "&" for command, so it stops if LVM command hangs.
 - Set core dump.
 - Run the script with 2 patterns.
 - Singly
 - Parallel
 - Problem is reproduced when script runs in parallel.
 - Demonstrates that the problem is an LVM specification for that version.
 - Change operation to not run LVM command in parallel to ensure your understanding of the problem is correct.

Set tools to gather informations

The tools 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 isn't found, set monitor tools for all related components not only for gathering informations. (application, middleware, OS, system resource, performance) and if problem recurs, collect the data that is related to the problem.

1.3 Problem resolution and follow-up actions

After determining the problem you will have to fix the problem. Applying a patch, changing a configuration and operation may cause side effects. Before applying patches or making configuration changes, confirm the problem will be fixed with it. Confirm that there will be no side effects. Ensure that you update documentation.

Before applying a fix(patch or change), a pre-apply test is recommended for confirming the problem will be fixed with no side effects. This test should include a test to try and reproduce the problem. If the problem didn't happen frequently you will need to determine how long to wait for the problem to recur to ensure 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, re-production test should show a clear result.

1.3.1 Expanded check for other systems

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

The following list are reminders of what to watch for on your other systems:

- ▶ Problem conditions. You will want to check to see if there are any other conditions which cause the same problem. Check the:
 - Software version
 - Transaction type
 - Operation
 - Configuration
- ▶ System scan
 - Scan all systems which 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 possibility is low, schedule the application of the fix for the next maintenance cycle or determine if 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 of 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 if there is additional system loading, you should make sure you document discuss any new threshold values for the system.

Additionally, update for:

- ▶ New message or changing monitored message
 - Add or change target message
 - TEC, Console, AOEMF, IBM Director, etc.
 - Add or change messages and code manuals
- ▶ Additional system loading
 - Change threshold value
 - Add new monitoring contents
 - New fixed system may change needless monitoring contents to needed one.

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 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 correct procedure?
 - Is there any time loss for first actions?

- ▶ Gathering Information
 - Are there any lack for PD informations?
 - Is it enough resource for gathering informations?
 - Operation persons
 - Automation scripts
 - System resources
- ▶ Reporting
 - Are all reports have correct informations related on each report line?
 - Lack of informations
 - Unnecessary informations
 - Is reporting time correct?
 - Is contact timing for support team correct?

1.4 Summary

In this chapter, we describe 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 would be to gather as much information as possible about the circumstances surrounding the problem. You will want to write as much information down about the problem as possible. You will want to examine if the problem occurs on just one system or many of your systems. You will want to determine if a change you or somebody else has made to the system happened just before the problem started.

Additionally, you will want to examine the frequency of the problem occurring and whether or not it is a problem that can be reproduced. You will want a test system that is set up as close to the production system as possible so you can try and reproduce the problem there.

In this IBM Redbooks publication, we continue to examine problem determination methods and the tools you would use 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 will 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, it is recommended that you use z/VM based tools or a combination of z/VM and Linux based tools to get more granular information on the problem. On the other hand, Linux based tools are often your best choices for debugging application, kernel or process related problems.

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

2.1.1 Collecting technical setup information

You will want to collect some technical setup information as part of your information gathering technique. You will want to gather your release and service level information, so you will use the command, **q cplevel** as shown in Figure 2-1 on page 22.

```
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 will 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 on page 23)
 - **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 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 userid.

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 on page 24.

Note: **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 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 on page 25 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
FTPSEVERE 000000:001412 TCPIP    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** (see Figure 2-5 on page 26) command to display, in order of their priority, current members of the dispatch and eligible lists. Issues 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 userid indicates the list status of the userid.

- ▶ 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 the command can be found in the manual z/VM CP Commands and Utilities Reference, SC24-6081-05

2.1.3 query srm

To get an overview about 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 on page 27 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 how to set and adjust the srm parameters see also Linux on IBM System z: Performance Measurement and Tuning, SG24-6926-01.

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 on page 28 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 Chapter 3.2.2, “When should a Trace be run?” on page 79, 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 show you an example. To interpret 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 setup 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.

The following are the steps to perform after you have decided to use trace:

1. Choose an event (or events) that you want to trace. You must decide which events 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 on page 29. 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 on page 29. 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 on page 30. 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 , TCP/IP - DSC , DTCVSW2 - DSC
DTCVSW1 - DSC , VMSERV - DSC , VMSERVU - DSC , VMSERVS - DSC
OPERSYMP - DSC , DISKACNT - DSC , EREP - DSC , USER12 -L0003
VSM - TCP/IP
[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

In parallel on the 3270 console the trace messages are displayed. See some of them in Figure 2-9 on page 31.

```

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 ODE1 -> 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 on page 32. 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/and sent to another system.

```

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 please also 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, you should 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 on page 33


```
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 on page 34 we see on the 3270 console of our Linux system, LNXIHS, the kernel panic messages. Now the **vmddump** 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.
ITSO 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 on page 35 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 DUMpload 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 on page 36 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.

2.1.7 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

Modes of Operation

The modes of operation for Performance Toolkit include:

- ▶ Basic Command Mode, usually referred to as 'basic mode' is the normal operating mode which 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 on CPU, I/O and user performance collected by this machine. It allows real time performance analysis of z/VM systems.

- ▶ Remote Performance Monitoring Mode displays performance data collected by the performance monitors from another virtual machine on the same or other VM systems.
- ▶ 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).

The 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 screen similar to the following shown in Figure 2-15 on page 40, 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 which 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 screen

This initial screen 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 use in the 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 screen should look similar to the Figure 2-16 on page 41

FCX124 Performance Screen Selection (FL530) Perf. Monitor		
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. SP00L 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 ==> 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 the 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 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, plus controller functions. The 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 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 screens 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 which cannot be generated on the system (e.g. LPAR data on an unpartitioned system)

Selecting Performance Screens:

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

- ▶ entering the number of the selected performance data display on the command line, or
- ▶ moving the cursor to the selected display (note that it will move only to highlighted numbers if you use the tabulator keys), or
- ▶
- ▶ entering the 'short-path' command for the desired screen

and then pressing the ENTER key.

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

A good starting point 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 the number of users who are waiting for a resource like storage pages or I/O.

The screen displayed is shown in Figure 2-17 on page 43

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		User Status:		
VMDBKs in queue			0		0		1		6		# of logged on users 25		
VMDBKs loading			0		0		0		0		# of dialed users 0		
Eligible VMDBKs					0		0		0		# of active users 11		
El. VMDBKs loading					0		0		0		# of in-queue users 7		
Tot. WS (pages)			0		0		871		1653k		% in-Q users in PGWAIT 0		
Expansion factor					0		0		0		% in-Q users in IOWAIT 0		
85% elapsed time			4.712		.589		4.712		28.27		% elig. (resource wait) 0		
Transactions			Q-Disp		trivial		non-trv		User Extremes:				
Average users			.2		.1		.3		Max. CPU %		LXOR4 95.1		
Trans. per sec.			.7		.6		.1		Max. VECT %			
Av. time (sec)			.307		.258		3.061		Max. IO/sec		LXOR4 113		
UP trans. time					.258		3.061		Max. PGS/s		LXOR4 9.6		
MP trans. time					.000		.000		Max. RESPG		LXORINST 511413		
System ITR (trans. per sec. tot. CPU)							1.4		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 userid we display the 21. User resource usage page. This display is show in Figure 2-18 on page 44.

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,---	DOR	
DTCVSW1	.00	.000	.0000	.0	.0	.0	.0	.0	ESA,---	DOR	
DTCVSW2	.00	.000	.0000	.0	.0	.0	.0	.0	ESA,---	DOR	
EREP	0	0	0	...	0	0	0	0	0	0	ESA,---	DOR	
FTPSEVE	0	0	0	...	0	0	0	0	0	0	XC,---	DOR	
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,---	DOR	
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,---	DOR	
OPERATOR	0	0	0	...	0	0	0	0	0	0	ESA,---	DOR	
OPERSYMP	0	0	0	...	0	0	0	0	0	0	ESA,---	DOR	
PERFSVM	.01	.005	.004	1.3	.2	.2	.1	.0	.0	.1	ESA,---	DOR	
TCPIP	.01	.003	.002	1.5	.0	.0	.0	.0	.0	.0	ESA,---	DOR	
TCPMAINT	0	0	0	...	0	0	0	0	0	0	ESA,---	DOR	
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,---	DOR	
USER10	0	0	0	...	0	0	0	0	0	0	ESA,---	DOR	
USER15	0	0	0	...	0	0	0	0	0	0	ESA,---	DOR	
VMSEVR	0	0	0	...	0	0	0	0	0	0	ESA,---	DOR	

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 userid. To do so just place the cursor on the userid you want to have more information about and press enter. In our example we want to get more information about userid LXOR4, the screen is shown in Figure 2-19 on page 45.

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 userid 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 userid LXOR4. We can therefore identify possible problems like looping, high i/o rates and so on. We can get more detailed information right out of this screen about the users DASD. To do this, place the cursor in the appropriate field in the screen and press enter. In our

example, we want to get more information about the DASD with the CHPID 0201. The result is shown in Figure 2-20 on page 47.

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 :	...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				
+-----				
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 userid 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 **redisp** or simply press the PF2 key from the IBM z/VM Performance Toolkit main menu. The screen displayed looks similar to our Figure 2-21 on page 49. Every 60 seconds a new set of variables are 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 if the system is in the an emergency scanning mode. The value can be between 0 and 100%. (Percent of demand scans which did not complete with scan 1). We discuss a scenario where emergency scanning occurs in Chapter 12.

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*

2.1.8 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 on page 51 demonstrates a graph of CPU utilization displayed by OMEGAMON.

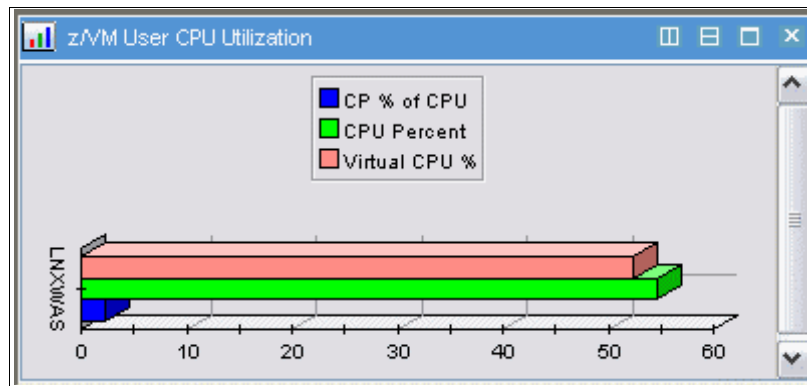


Figure 2-22 OMEGAMON CPU utilization graph

Figure 2-23 on page 51 demonstrates a graph of device utilization displayed by OMEGAMON.

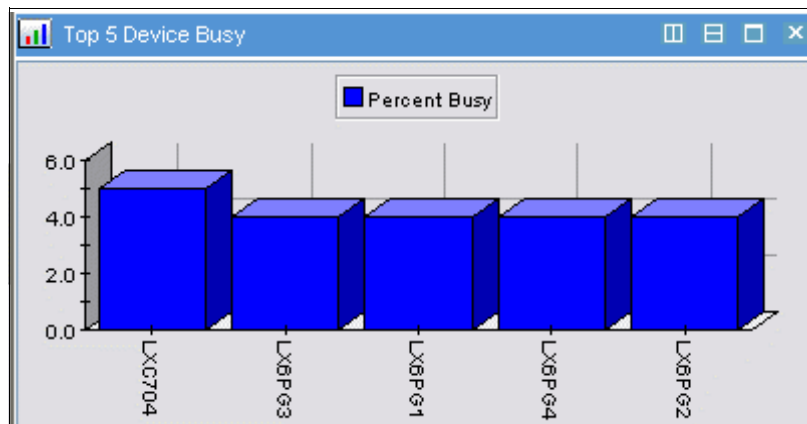


Figure 2-23 OMEGAMON device utilization graph

Figure 2-24 on page 52 demonstrates a graph of storage utilization displayed by OMEGAMON.

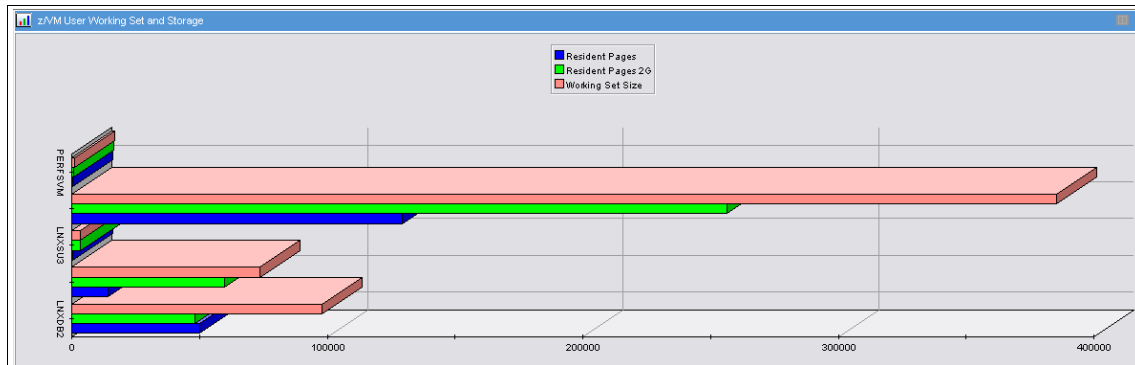


Figure 2-24 OMEGAMON storage utilization graph

The same data can be displayed from the OMEGAMON portal in text or tabular form through options in 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 on page 53 from OMEGAMON shows the data from our total hardware system.

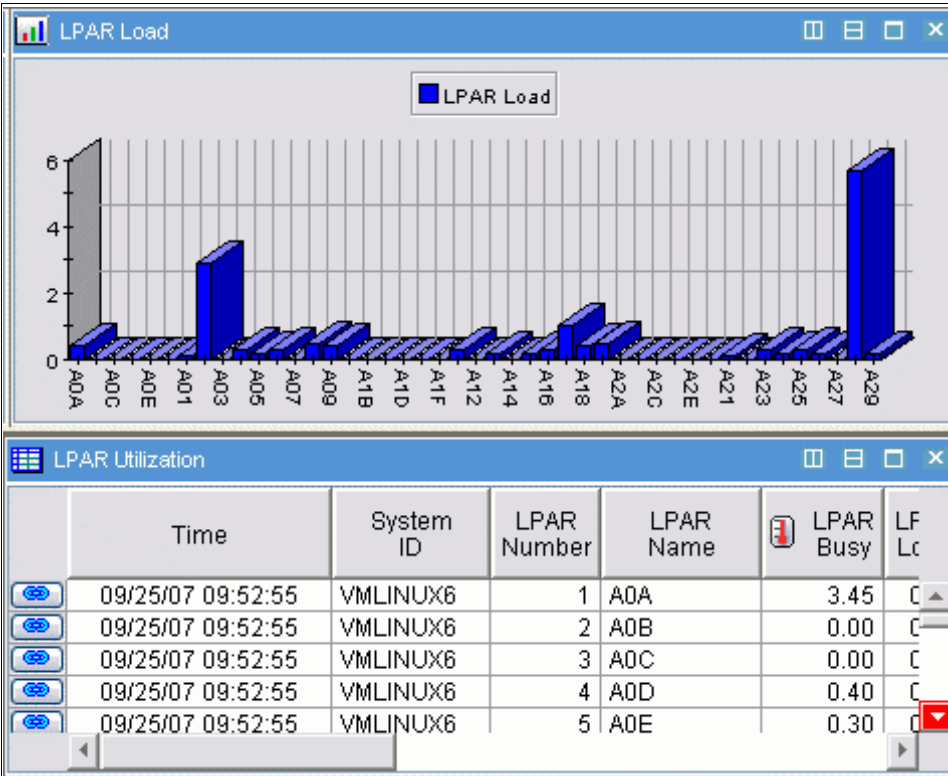


Figure 2-25 OMEGAMON LPAR utilization graph

The LPAR utilization shown in Figure 2-25 on page 53 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 the Figure 2-26 on page 54, 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).

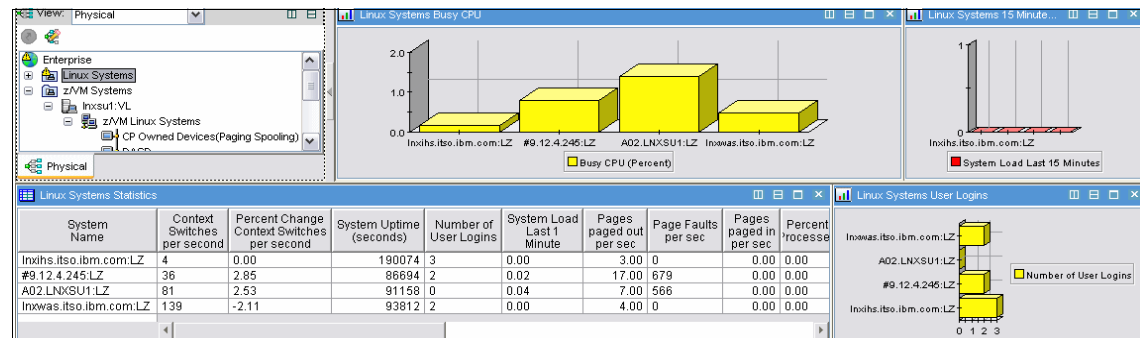


Figure 2-26 OMEGAMON Linux guest performance overall view

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 on page 54

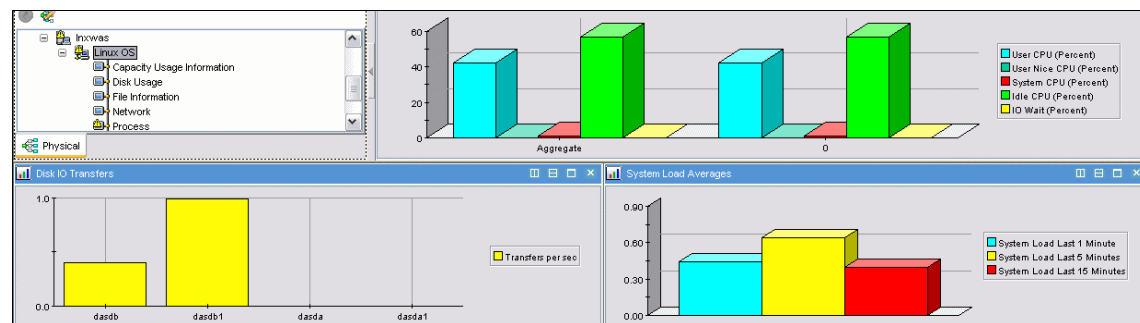


Figure 2-27 OMEGAMON Linux system graph

and Figure 2-28 on page 54.

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	LNxWAS	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 on page 55 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 on page 56 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.

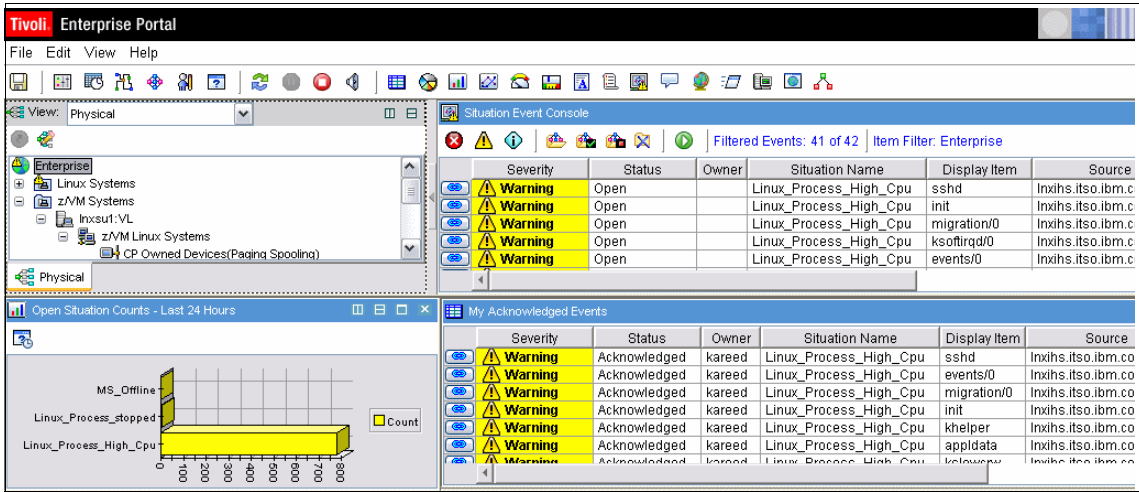


Figure 2-30 OMEGAMON situation alerts

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 determine which tools will be used to help determine the problem. It is always recommended to use 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. Here, we explore the information gathered and the circumstances where the tools would be useful for problem determination.

3.1 Information gathering tools

As discussed in the previous chapter, the first and foremost thing 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 they got in z/VM. It is highly recommended 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 which only provide snapshot utilization. So in a problem determination approach, it is recommended to use shell scripts to record the utilization and timestamp 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 on 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 filesystem
[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

```
lrx02:/ # 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® website.
<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 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 on page 60, 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 have to 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 on page 61, 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

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
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
18:56:04 dev94-4  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 backend 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 would analyze 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   kB/s   kB/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 activities among all of the processors are also reported.² **mpstat** displays more in-depth CPU statistics than **iostat**.

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

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

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
```

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 sleeping. Information is updated every three seconds, or at a chosen interval. It 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, top command includes a new field “CPU steal time” – time Linux wanted to run, but z/VM gave the CPU to some other guest. This field will be very useful to understand CPU performance characteristics from within Linux. Also its worth to note that top is a resource consuming tool, which means 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 on page 64, 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 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 which can be used with various combinations of parameter to obtain relevant information required on the processes. For more information on the various parameters, please refer to the respective `ps` man pages.

3.1.6 ipcs

The **ipcs** command reports information about active interprocess communication facilities. The report would have information on currently active message queues, shared memory segments, semaphores. `ipcs` command is a very powerful tool which provides a peek 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 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 on page 66, there are lots of blocks being read and written to the `dasd`'s. 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 filesystem. 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 systems dasd statistics. By setting off and back on the dasd statistic option we can resets 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 was 56881.

Line 2 contains the number of 512B-blocks transferred during the observation period, in this example it was 5270816 which is approximately 2.5GB.

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

Each line represents a histogram of times for a certain operation. Operations split up into the following:

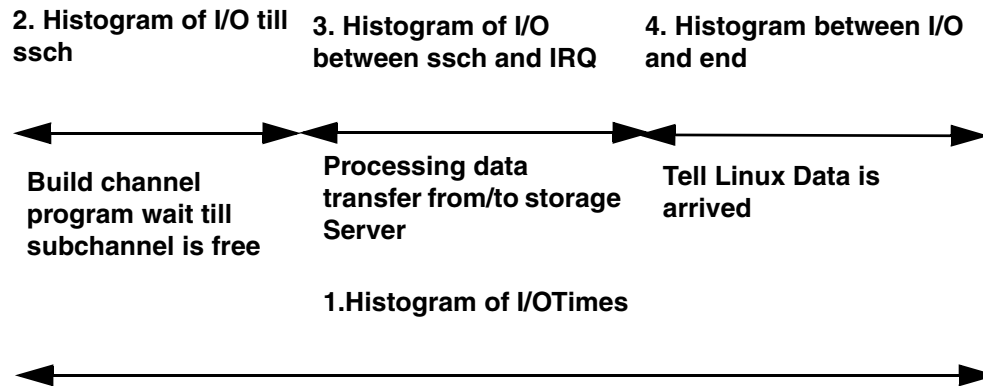


Figure 3-1 Explanation for the dasd statistics Histogram details

- 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 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), i.e. 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 on a particular dasd, we can use the **tunedasd** tool.

In Example 3-15, we issue the **tunedasd** command against our dasd where the root filesystem is mounted (i.e. /dev/dasdb1).

Example 3-15 Gathering I/O information on 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	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	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	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	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	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	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	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 RedHat 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 above. For more information on 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 doesn't 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 (shutdown) 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 setup. Kernel or user space code is profiled based on these settings.

Example 3-19 on page 72, 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 above. 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 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:~ # oprofilet --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 on page 74 for an overview of zFCP statistics.

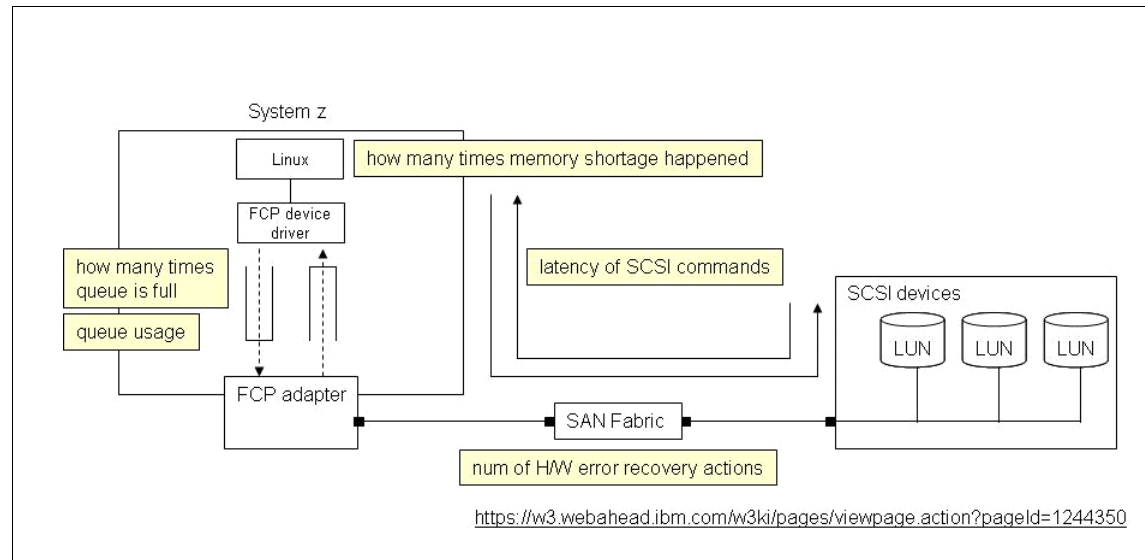


Figure 3-2 zFCP statistics overview

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 higher) and SUSE Linux Enterprise Server 10 GA (kernel version 2.6.16.21-0.8 and higher).

Gathering zFCP statistics

The following are the steps 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/.
2. For SLES9, depending on where proc is mounted (SLES9 does not use debugfs): <mount_point_proc>/statistics.
3. 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.
4. 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
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 on page 76.

zFCP Statistics

Breakdown of SCSI latency that can be acquired

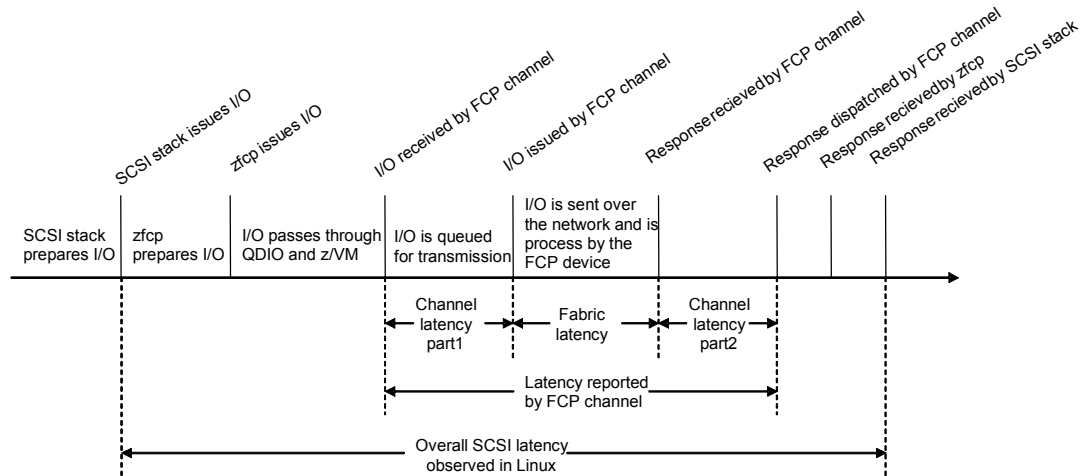


Figure 3-3 SCSI latency breakdown

A sample of SCSI latency statistics can be seen in Example 3-24 on page 76.

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, a summary of tools you can use in your Linux environment to gather information is provided in 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		

Tools	Description	CPU	Memory	Disk	Network
iostat	Provides information on 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 dump technique can only take snapshot of the kernel structure at a given point of time.

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

3.2.1 When to take Dump?

When an application fails and halts execution or when ever a system crashes with a kernel panic, It is recommended to take a dump. Since with a dump we can record and analyze the execution and memory snapshot of the system when the issue happen.

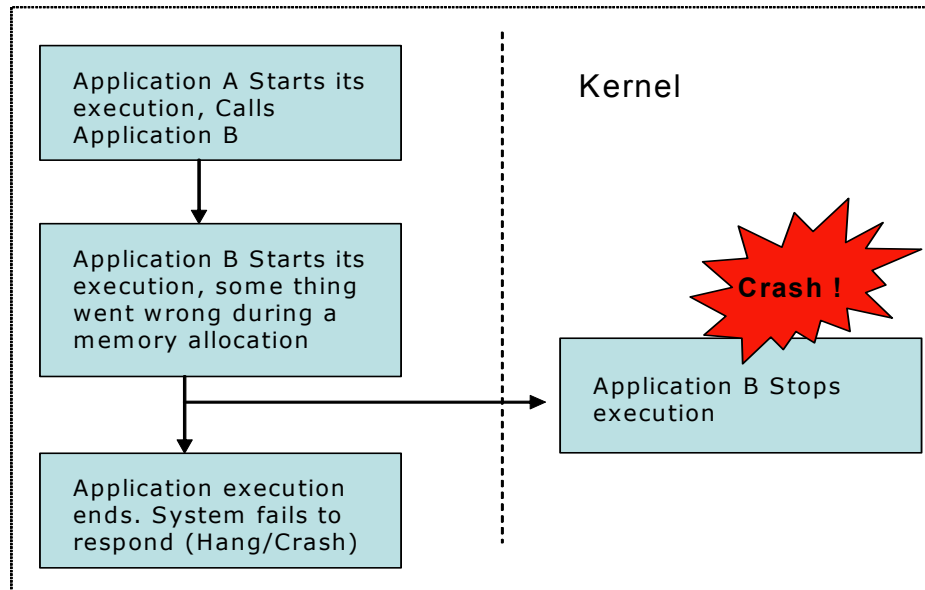


Figure 3-4 Example of when to use dumps

3.2.2 When should a Trace be run?

Unlike the Example 3-4 on page 79, there are situations where the application would be still in execution, but doesn't respond. In this case we would its recommended to use Trace facilities, rather than the dump.

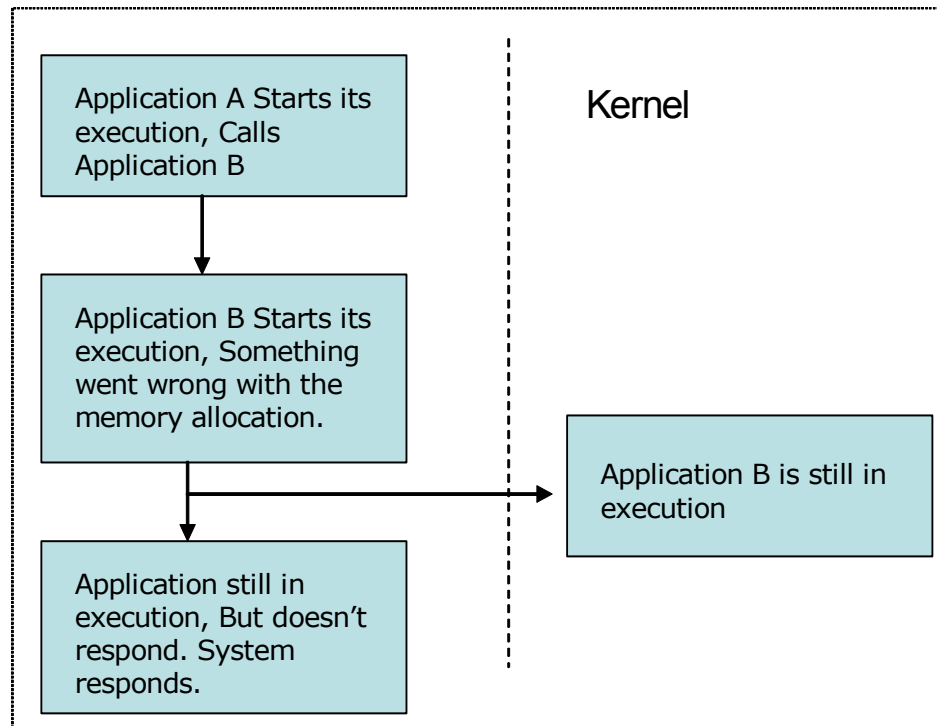


Figure 3-5 Example of when to use Trace Facilities

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

3.2.3 Standalone Dump

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

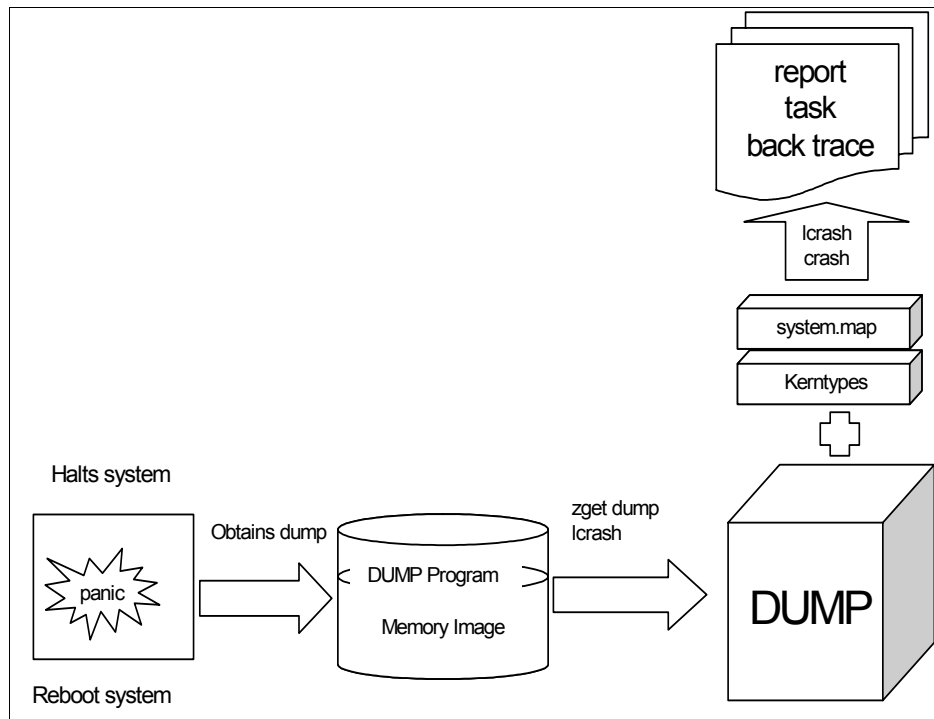


Figure 3-6 Standalone dump

Three standalone dump tools are shipped in the s390-tools package as part of the zipl 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

Steps to create a standalone dump:

- ▶ Create a dump device for dumping the memory. It is recommended that you have device storage allocation around Memory Size plus approx 10 MB space. The zipl.conf .
- ▶ To initiate a dump, we must IPL the dump device. This is destructive, that is, the running Linux operating system is killed.
- ▶ 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 couple of tools to validate the dump devices or files.

zgetdump

Use `zgetdump` tool to read the the given dump device and write its contents to standard output or redirected 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 analyser.

Example 3-26 Verification of the dump file using lcrash

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

To convert a dump image to a file using `lcrash`, we need to issue the f

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 a 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 following commands

Example 3-28

```
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 on page 83, 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

From Figure 3-7 on page 83, the summary says that system crashed because of a software failure. Like wise the report consist of other information as listed above, which can be useful for problem determination at a high level.

=====							
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

Figure 3-8 on page 84, shows the task list of an lcrash report. This can used to find out all the process that are running during that point of time.

Also the log buffer would provide use with useful information on the log messages that dont appear on the console or dmesg. But its is recommended to send 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 distribution, core dumps are disabled. Once we have the core dump

enabled in the system, the next time some application crashes, Linux would be able to write the core to a file.

As an example, we executed a simple program which 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 get finished so that it can span more process.

It is an classic example for an 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 of the the Application is not responding and the Linux system is also low on resources.

In Figure 3-9 on page 85, 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`, `strace`.

```
lnx02~ # 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 process 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` as well. 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 setup 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 `/etc` directory.

Example 3-29 example to setup 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 are some of the important signals available in Linux. We can use any of the Signal which has an action of creating a Core dump.

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

Here in Example 3-30 on page 87, we would issue the SIGABRT (Abort Signal) signal for obtaining a core dump.

Example 3-30 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 need 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 Example 3-11 on page 88, gdb have 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
0x000000200000cc1d4 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 Redhat 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 3-31 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 would able:

- To analyze kernel panic, Oops message etc.
- To identify direct reasons to lead system down
- To identify what tasks are processed at failure time
- To 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.
- To analyse the memory sturctures 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..etc.

3.3 Debugging Tools

Linux based tools are best choices for debugging application, kernel or process related problems. In this section, we would 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 appropriately.

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 problem cause can be narrowed down. Also we would be able to immediately find out whether the application layer is causing the problem or not. If the application is causing the issue, using the print statements we would be able to narrow down the area of the application which 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 of Break points
- Displaying of 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 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:

- To View the running process using attach command.
- To Start the program using run command.
- To use the core file to view the state of the process during it crashed.

In Example 3-32 on page 90, we executed a simple program which does an illegal memory reference and fails with a segmentation fault.

Example 3-32 program execution which fails with a segmentation fault

```

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

```

Now we will load the is executed which would with gdb and analyse through few options in gdb. In the Example 3-33 on page 91, when the program receives a segmentation fault, gdb returns to its prompt. Also lists the place where the segmentation fault occurs with the line number.

As closer look would reveal that the program is trying to initialize or read a unknown memory location. Also from the gdb output its clear that the info locals and list commands get information about the currently selected stack frame.

Example 3-33 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:~ #
```

Analysing a Core file using gdb

As already discussed in the previous section, we obtained a core dump for the segfault program.

Example 3-34 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 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 which had been allocated to it.

Now we will 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:

Example 3-35 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 3-35 on page 92, gdb when it tries to print the variable, it displays a message that its not able to access the memory location. And thats the reason for the segmentation fault.

Note: For more indepth information on gdb, please 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 which are called by a process and the signals which 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 debuggin in two methods:

- ▶ Starting the Program or Application with the strace utility

Example 3-36 example for obtaining strace for ls command

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

- ▶ We can also obtain strace of a process which is already running.

Example 3-37 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 certain option. But it is always recommended to obtain the traces of all system calls when the problem area is not narrowed down. In Example 3-38 on page 93, strace would only trace the signal system calls for the process.

Example 3-38 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 open() system call, using the option “-e trace=open.

For our demonstrating of the strace, we wrote a simple program which will try to open a file descriptor which is not been initialized at all. Even though the program doesn't fail, using strace we would be debug that the program has some flaws. In Example 3-39 on page 94, strace points out that the program tries to open a file which is not available at all.

Example 3-39 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\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\1GNU\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 which are called by the executed process and the signals which 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 much similar in obtaining the traces. The ltrace utility can be used for debugging in two methods:

- Invoking the user program or application with the ltrace utility itself

Example 3-40 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) +++
```

► We can also hook the PID of a process already running with the ltrace utility.

Example 3-41 *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, 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 the Example 3-42, we have passed ls command as the executable to be traced. From the Example 3-42, we would be able find out, which function is called most of the time or consumes lots on CPU cycles. By this the developers can optimise there code to run efficiently.

Example 3-42 *Call tracing using ltrace*

```
lnxihs:~ # ltrace -c ls
autoinst.xml forker netperf out_file segfault tcp_crr tcp_stream
threader vpd.properties
bin kerntest netserver rmfpms segfault.c tcp_rr test.c
threader.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

It is a key combo we can hit which 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 system's would generate a /var/log/messages file for the back trace.

To enable the feature for the running kernel, issue the following commands. Here in Example 3-12, we list of 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 3-43 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 the information on using the Sysrq, we can issue “echo t > /proc/sysrq-trigger”. The kernel traces would be written to “/var/log/messages”

Example 3-44 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 0000000000035d1b8
0000000000012a652 00000000efc7ba60
Mar 25 13:54:34 lnx02 kernel:      000000000001950d0 00000000fc6e1d00
07000000efc7bb78 07000000efc7bb78
Mar 25 13:54:34 lnx02 kernel: Call Trace:
Mar 25 13:54:34 lnx02 kernel:  [<0000000000013db4a>] schedule_timeout+0x9e/0x150
Mar 25 13:54:34 lnx02 kernel:  [<000000000001b2080>] do_select+0x4a0/0x5e8
Mar 25 13:54:34 lnx02 kernel:  [<000000000001b2514>] sys_select+0x34c/0x588
Mar 25 13:54:34 lnx02 kernel:  [<0000000000011f2ac>] sysc_noemu+0x10/0x16
```

Sysrq Key assignments

Table 3-4

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 filesystems.
'u'	Will attempt to remount all mounted filesystems read-only

KEY	Operation
'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 info to your console.
'v'	Dumps Voyager™ SMP processor info to your console.
'e'	Send a SIGTERM to all processes, except for init.
'i'	Send a SIGKILL to all processes, except for init.
'I'	Send 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.)

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

Figure 3-13 Sysrq Log levels

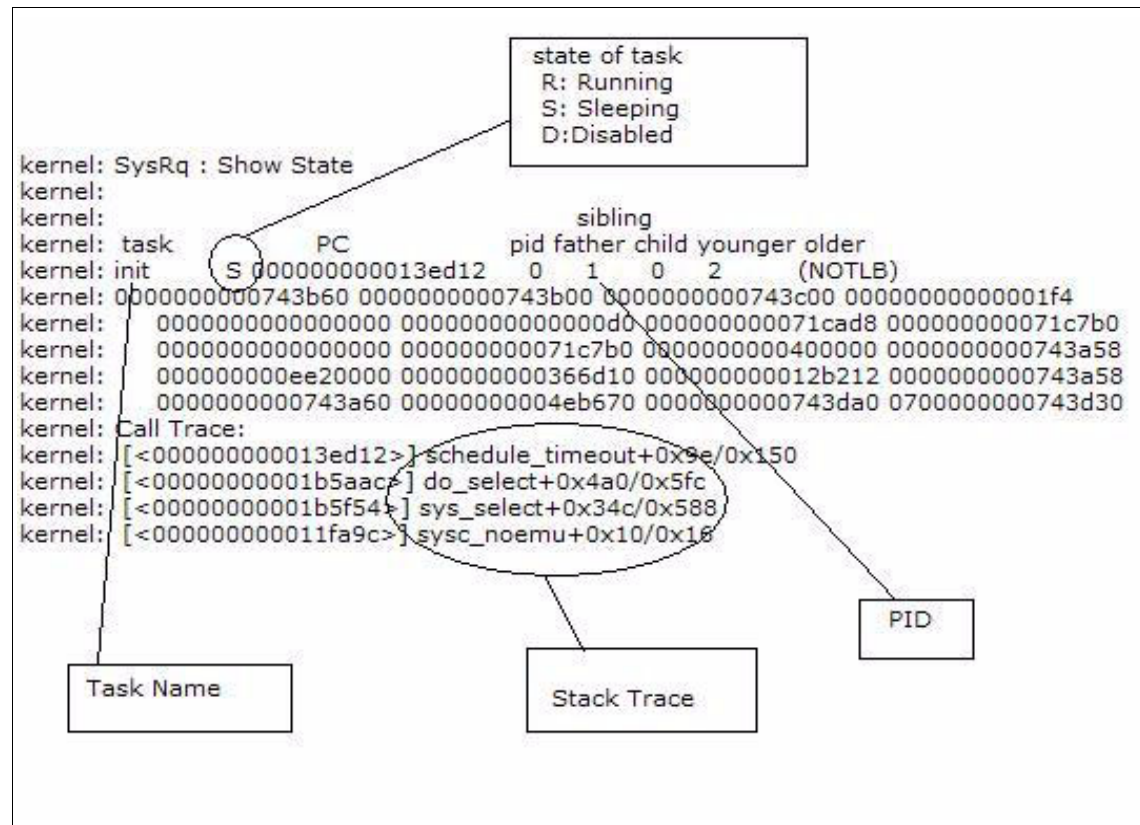
Note: There is some reported problems, which lead to a kernel panic when SysRq key is used. Kernel is been fixed and the problem doesnt appear on the release

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

Stack trace by SysRq is useful to analyse strange behaviour in the system, with multiple users space programs stops responding and when the key board input can be possible. Usually the SysRq trace would be used in conjunction with other Stand alone and core dump.

SysRq trace

Figure 3-14 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, select the file under /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 an actual debug level (range 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 which is provided for every debug log.

Example 3-45 s390dbf log level

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

Now according to the level of tracingrequired, set the the log value as below

Example 3-46 Setting up log level

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

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

Example 3-47 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 back to the old tracelevel.

Example 3-48 Resetting th tarce level back to default

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

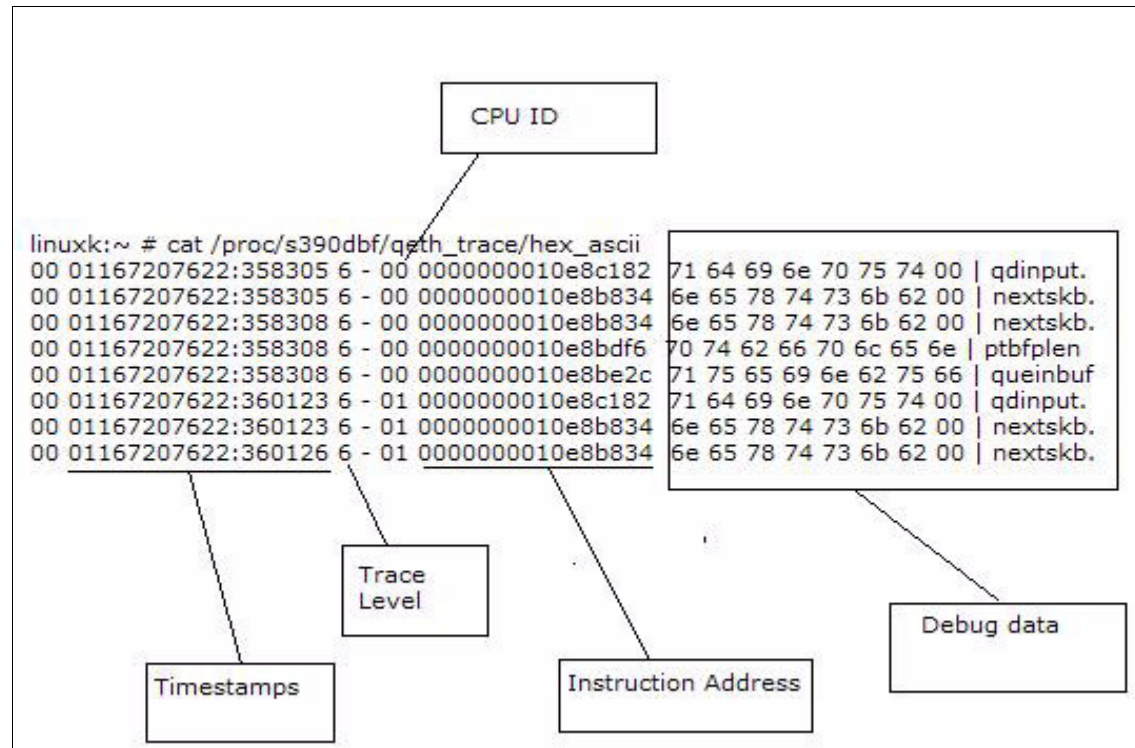


Figure 3-15 example s390dbf qeth_trace

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

3.3.6 zFCP/SCSI

zFCP trace feature enables the debugging of zfcps device driver and the flow of Fiber 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 the Example 3-49 example below, we are enabling zFCP driver with the log level of “loglevel_erp” to 1.

Example 3-49 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 information is obtained The log level of all area are set to 0 by default and only minimum information is output. The debug message is written in the /var/log/messages by setting log level of zfc. To stop the zFCP trace: see Example 3-50

Example 3-50 disabling zfc

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

Following is the Example 3-51, the output for the zFCP driver

Example 3-51 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. Following is the command, by which we can set the log level of SCSI trace.

Example 3-52 leg level setting of SCSI

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

Example 3-53, shows the a example SCSI trace

Example 3-53 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 occuring in the SCSI and zFCP areas. It is neccesary to be cautious on the target area and the loglevel of the traces. Since the more and more the loglevel there would be some performance impact.

4



Network problem determinationProblem Determination

This chapter discusses network problem determination for LinuxLinux on System
zSystem z. Like the other chapters in this book, the focus of this chapter is on
Linux running on z/VM.

In this chapter, the following topics are discussed:

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

4.1 networking option:OverviewOverview 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 systemoperating system can use any of the physical networking options as well. Thus Linux systemLinux systems running as virtual machine (VM)virtual machines 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:AdapterOpen Systems Adapters
 - Hipersockets
- ▶ Virtualization technology
 - Guest LANGuest LAN
 - z/VM virtual switch (vswitchVSwitch)

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). Depedning on the exact type of the OSA cardOSA card used, 10 Gigabit Ethernet (10G or 10 GbE), Gigabit Ethernet (GbE), 1000Base-tT Ethernet, Fast Ethernet (FENET), Token-Ring and Asynchronous Transfer Mode (Asynchronous Transfer Mode (ATM)ATM) are supported. For Linux connectivity, these cards are recognized by the hardware I/O configurationhardware I/O configuration as of of the follwoing channel types:

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

Note: For detailed description of OSA-Express operating modes and limitations, refer the IBM Redbook OSA-Express Implementation Guide, SG24-5948

Queued Direct I/O (QDIO) 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:microprocessorOSA card microprocessor and TCP/IPTCP/IP stack.

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

In non-qdio modeQDIO mode, the data follows longer I/O path. Linux uses the LCS device driverdevice driver to communicate with the device when it is running in this mode. Figure 4-1 on page 109 shows the difference in the QDIO and non-QDIO data paths.

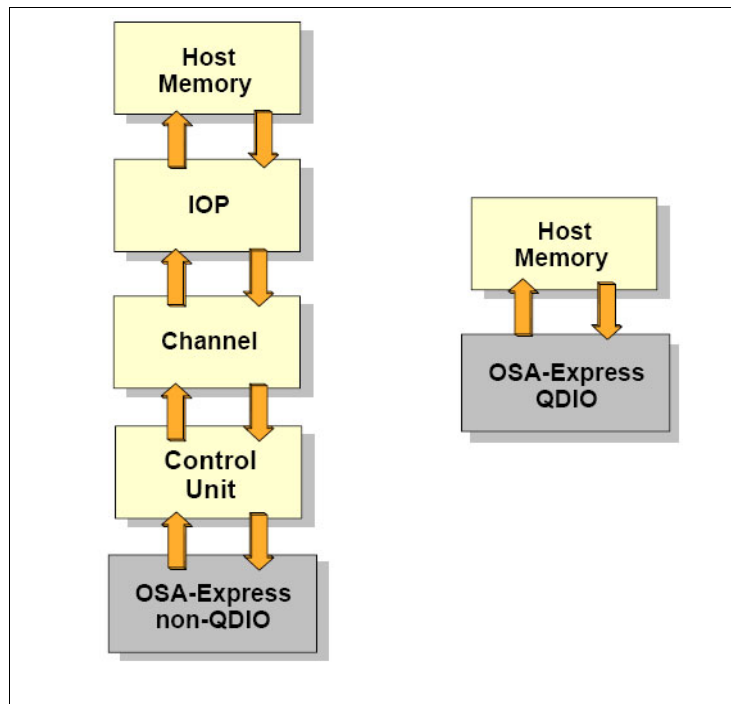


Figure 4-1 QDIO and non-QDIO data paths

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 layerIP layer). Each mode processes network data differently.

Layer 2 has three characteristics:

- ▶ Uses the MAC destination address to identify hosts and send and received Ethernet frameEthernet 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 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 addressIP 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 modeQDIO mode has the ability to send and receive IP traffic between LPARs without sending the IP packetIP packets over the network.

For outbound packets, OSA card uses the next hope address provided by the TCP/IP stack to determine where to send the packet. If this next-hope 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 make 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:QDIO interface OSA card QDIO interface.

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)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 on page 112, 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.

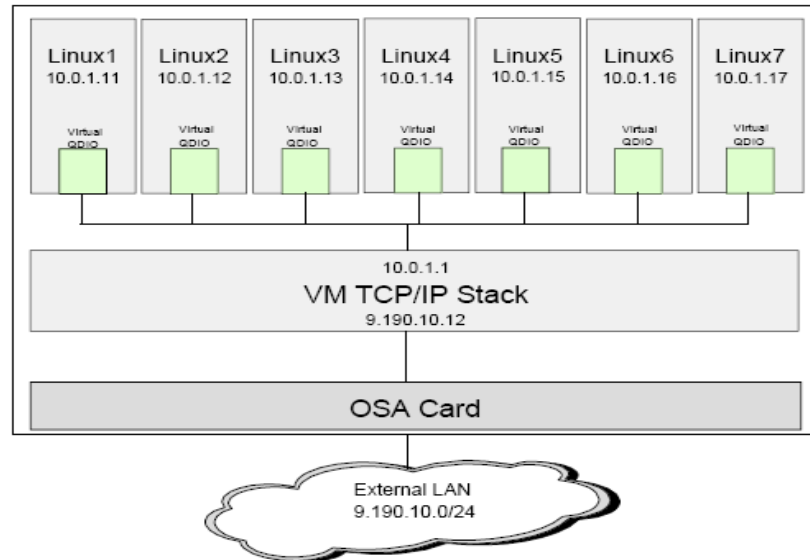


Figure 4-2 z/VM Guest LAN

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 OSA card:port OSA card port. Two additional OSA card ports can be specified as backups in the VSWITCH definition. The Linux guest Linux guest connected to the VSWITCH are on the same subnet as the OSA card port or 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 networking option:SummarySummary of networking options for Linux on System z

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

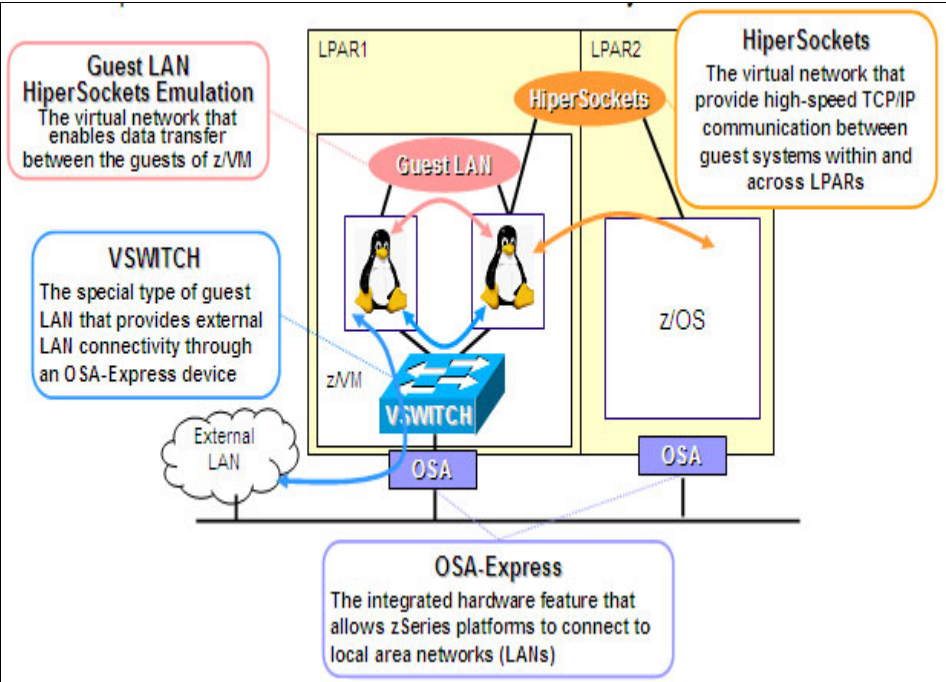


Figure 4-3 Summary of important networking options for Linux on System z

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 file Configuration files and boot or command execution messages from SuSE Linux:Enterprise Server SuSE Linux Enterprise Server will be used as examples throughout this chapter. However, the concept applies to RedHat 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 which 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 on page 114 lists these characters with their meaning.

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

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

4.2.2 Flow of network interface detection and activation under Linux

Figure 4-4 on page 115 summarizes the network interface detection for Linux on System z. The hotplug or the coldplug scripts ensure the right modules are loaded with the right options. Then the interface name interface names are allocated. Once the interface names are allocated, appropriate configuration files are read and network parameters are configured.

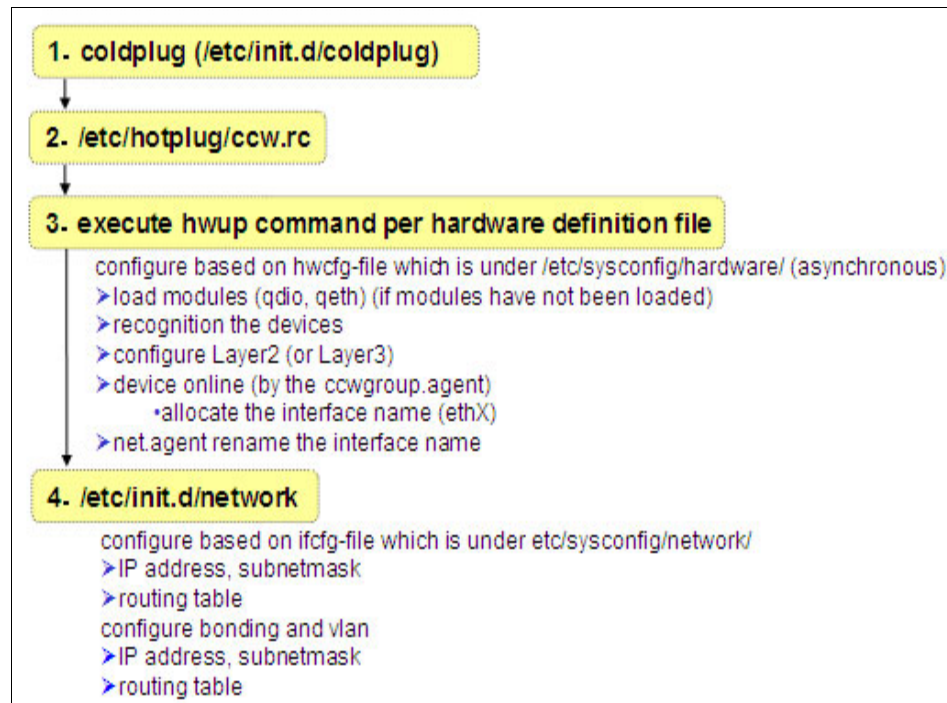


Figure 4-4 Linux network interface detection and activation flow

Figure 4-5 on page 116 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 **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 interface name. Once the interface names are allocated, then 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.

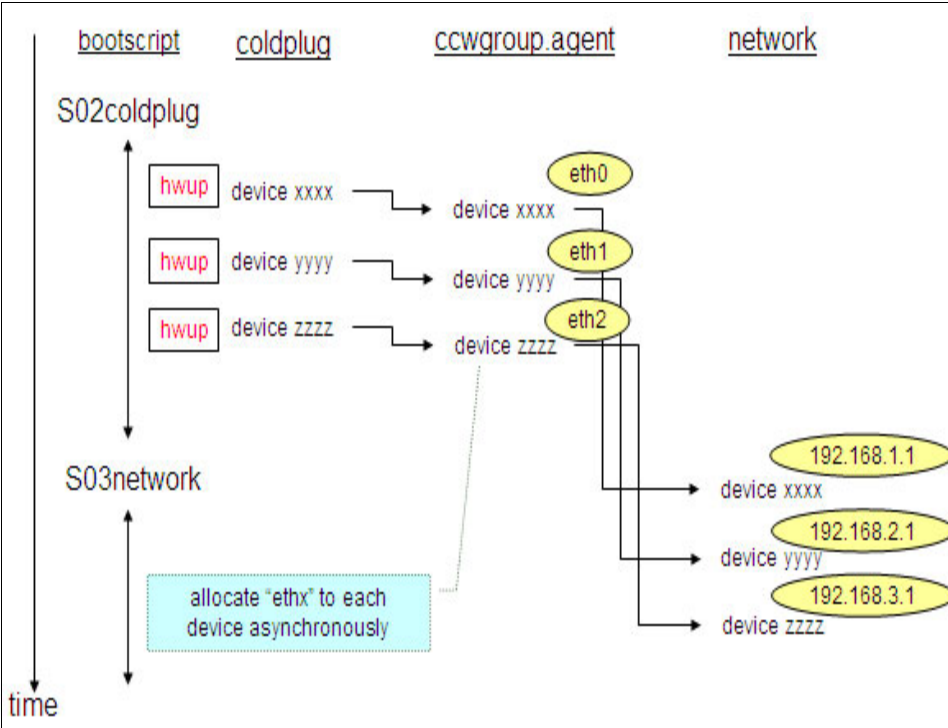
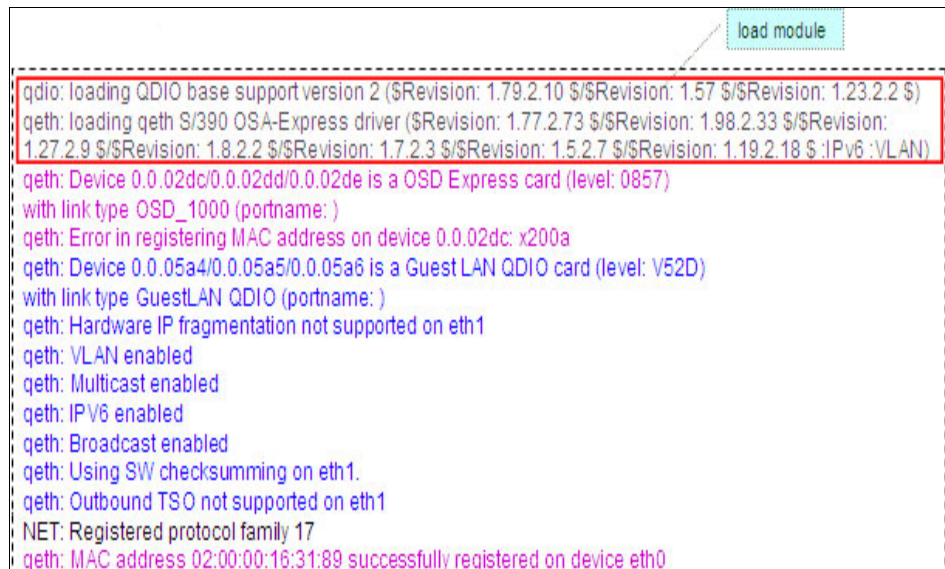


Figure 4-5 Linux network interface detection sequence

Figure 4-6 on page 117 shows sample boot messages generated by a Linux systemLinux 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 QDIO mode:network devicethe network device in the QDIO mode. The next lines indicate the details of the device detected. In Figure 4-6 on page 117, the fourth line from top indicate that the device addresses are 0.0.02dc, 0.0.02dd, and 0.0.02de. OSD in the same message line indicate 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.



```

load module
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 on page 118 indicate the boot messages related to a layer-3 configuration as well as those for a layer-2 configurationlayer-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 layer3 and layer2 configurations

4.2.3 Network interfaces and their interface names

On systems with multiple network interfaces, the interface name allocated to the OSA device OSA devices may not be consistent across Linux system:reboots 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 middleware certain middleware such as Oracle®. Also, SNMP and the Linux monitoring tool sar may also get affected by this issue.

Figure 4-8 on page 119 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.

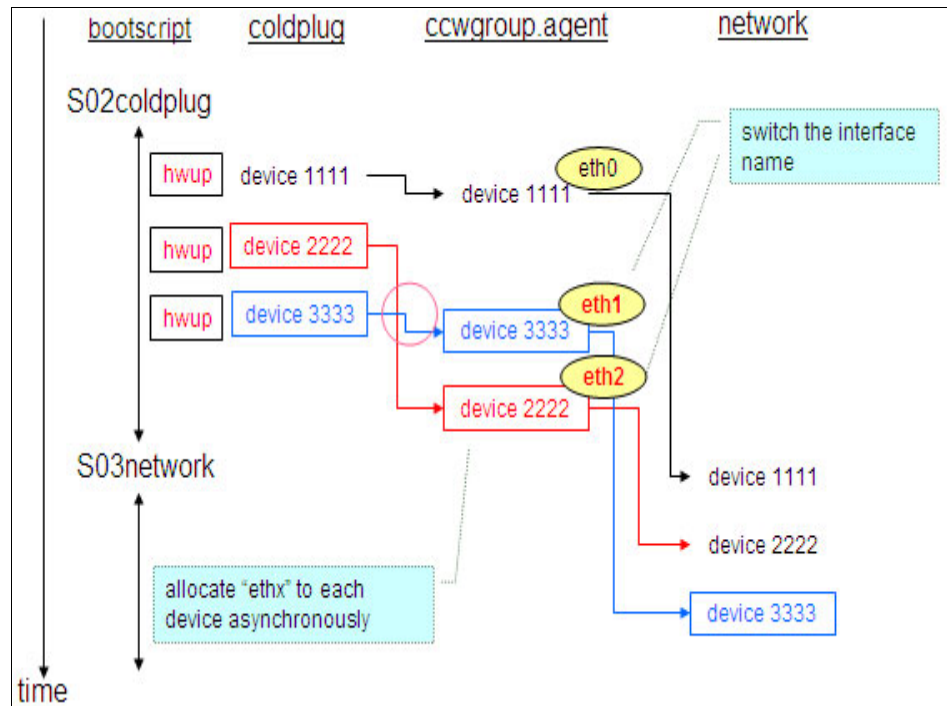


Figure 4-8 Asynchronous hotplug events and inconsistent interface naming

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 on page 119 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 will take a quick look at some of the configuration file important configuration files for network devices on a Linux system running on System z. As noted earlier, SuSE Linux:Enterprise Server file 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 on page 120 shows a sample hwcfg-qeth-ccw-0.0.xxxx file. For each OSA device OSA device on the system, there will be a unique hwcfg-qeth file. This device in this case is 0.0.02d4 and hence the exact file name is hwcfg-qeth-ccw-0.0.02d4.

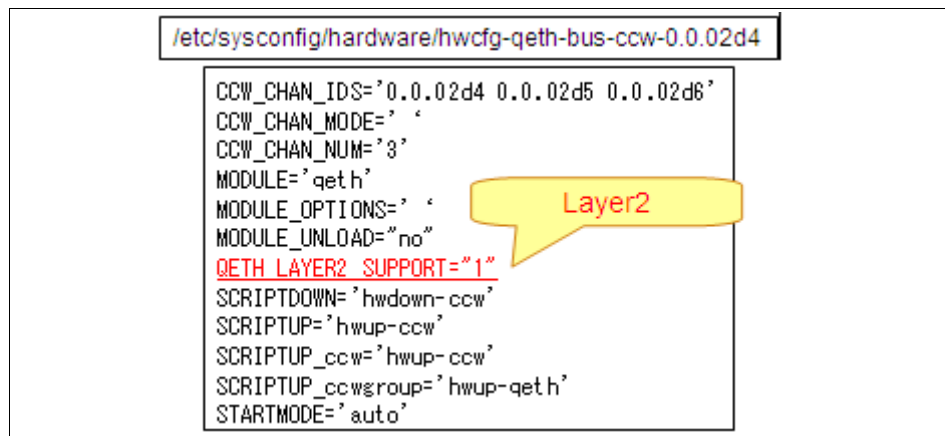


Figure 4-9 Network device is defined in /etc/sysconfig/hardware/hwcfg-qeth-ccw-xx

/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 121 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. IP address, subnet mask, local MAC address, MTU, automatic startup (STARTMODE), etc are configured in this file. When STARTMODE=onboot, the interface is activated, automatically, during the system startup.

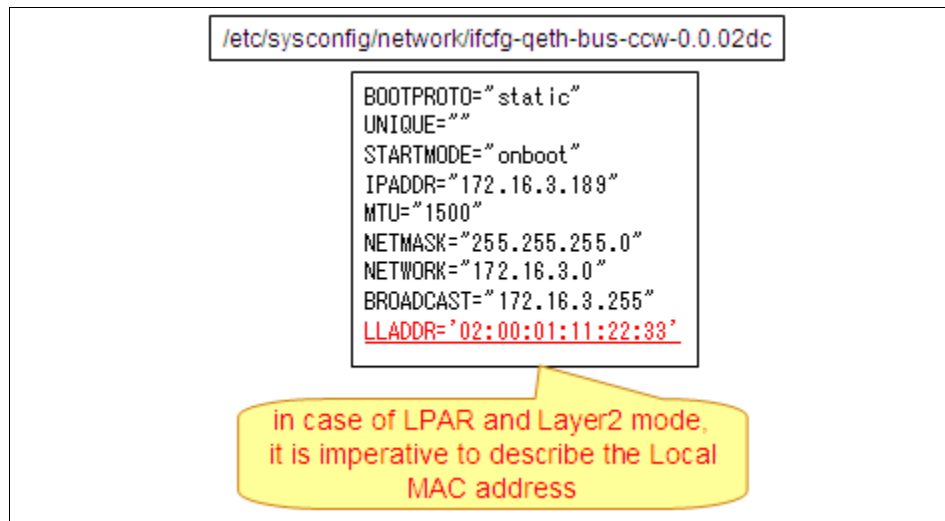


Figure 4-10 IP address is configured in `/etc/sysconfig/network/ifcfg-qeth-bus-ccw-xxx`

`/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 on page 121 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 on page 121 is a special entry which defines the default gateway. The keyword “default” identify it as the default gateway. The two minus signs (-) indicate that they assume the default values.

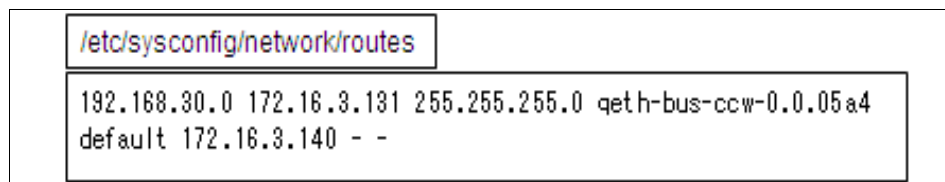


Figure 4-11 Network routes are configured in `/etc/sysconfig/network/routes`

4.3 The art of 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 the art of 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 will be discussed in detail in this chapter. Chapter 5, “Performance Problem Determination” on page 157, is dedicated to Performance Problem Determination 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, network communication problem on a system can be a problem local to that system or it can be outside (that is the local machine network is functioning correctly) of that system. 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 to other systems, there are some logical questions 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? 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 123 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) 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 laer. Of course, the IP address gets mapped to the IP layer and all TCP/UDP related stuff to the TCP/UDP layer TCP/UDP layer.

Finally, applications such as SSH, FTP, Telnet, etc are represented by the Application layer.

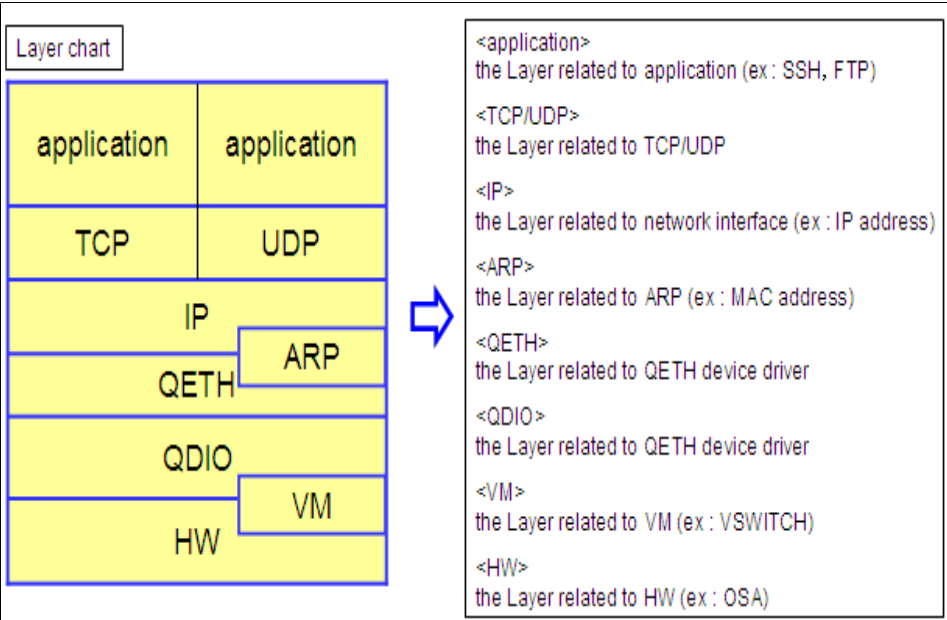


Figure 4-12 Network communication layer for Linux on System z

Figure 4-13 on page 124 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. Data link layer of OSI model is represented by the ARP layer. Also, the ARP layer partially along with the IP layer makes up the network layer of OSI. 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 layer of the OSI reference model.

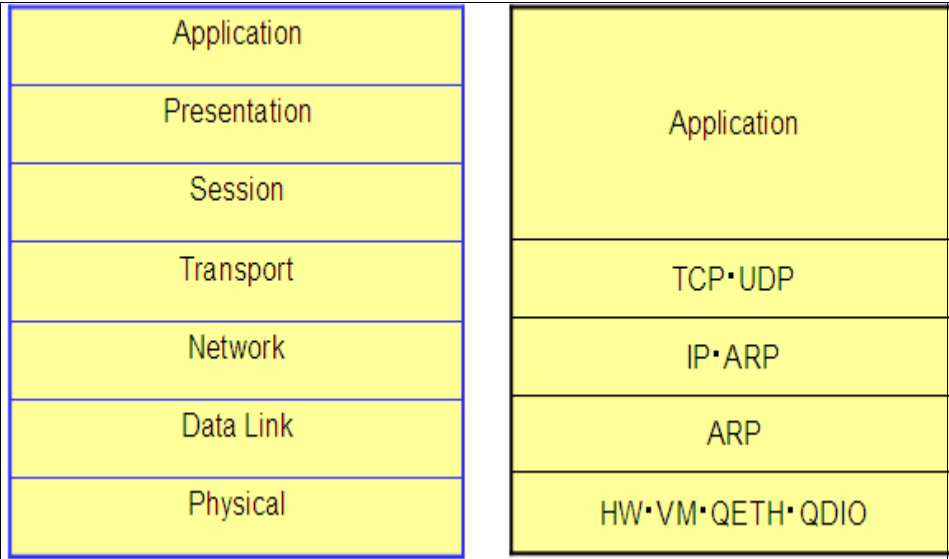


Figure 4-13 Network communication layer for Linux on System z and the OSI reference model

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 125 shows a high level flow chart demonstrating how this can be done. This flow chart describes an approach which 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. Some cases it may happen that the error messageerror messages gets 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, etc, the important thing to investigate whether any chages were made on the system before the problem occured. Most of the cases, we will notice that there was a change made and it resulted in an error. If the problem is occuring 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 problem 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, it is easy to identify and fix compared to the case of an intermittent problem. Intermittent problems may take more time

to troubleshoot and also, may need to be kept under observation for sometime after fixing to ensure that it is really fixed.

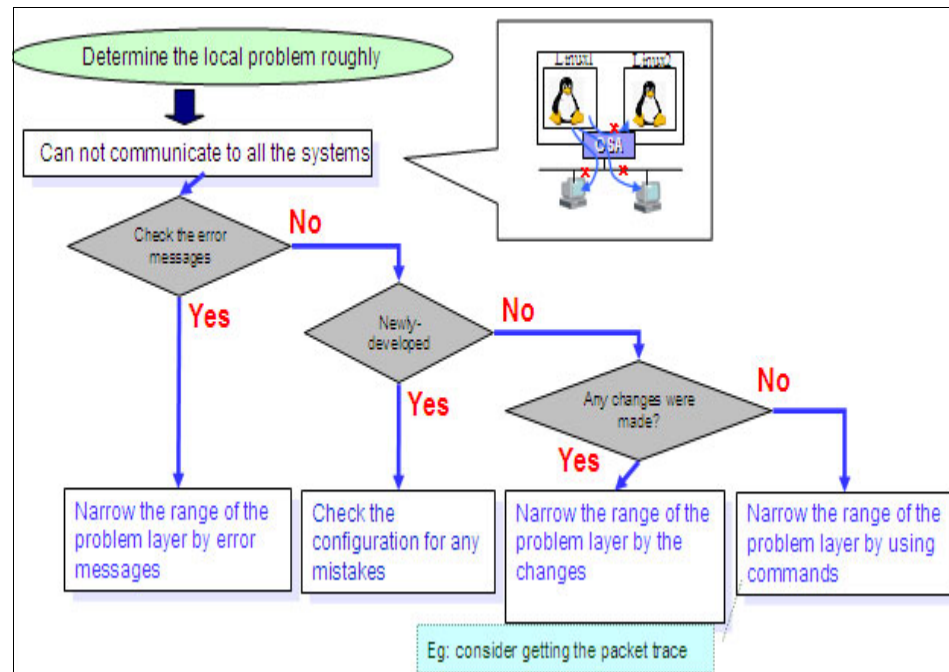


Figure 4-14 Problem determination on the local system

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 on page 126 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 success, then we are sure that the network cable is connected, network interface is active and IP communication (network layer) is happening. In such a case, it is certain that the problem is some where above the IP layer

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 cable not connected, etc. In this case, it is clear that the first level troubleshooting should start at the IP layer level and below.

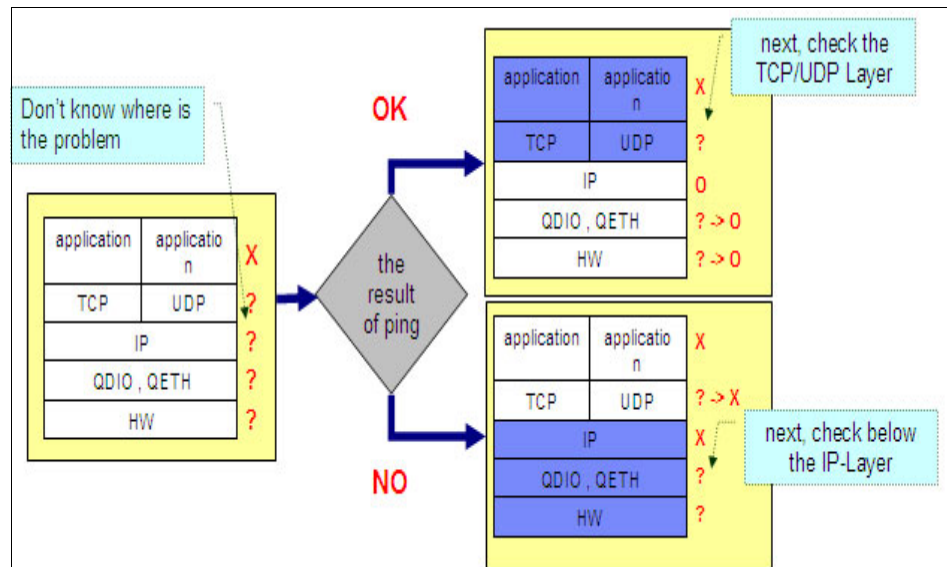


Figure 4-15 Using the ping command to narrow down to the problem layers

We will 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.

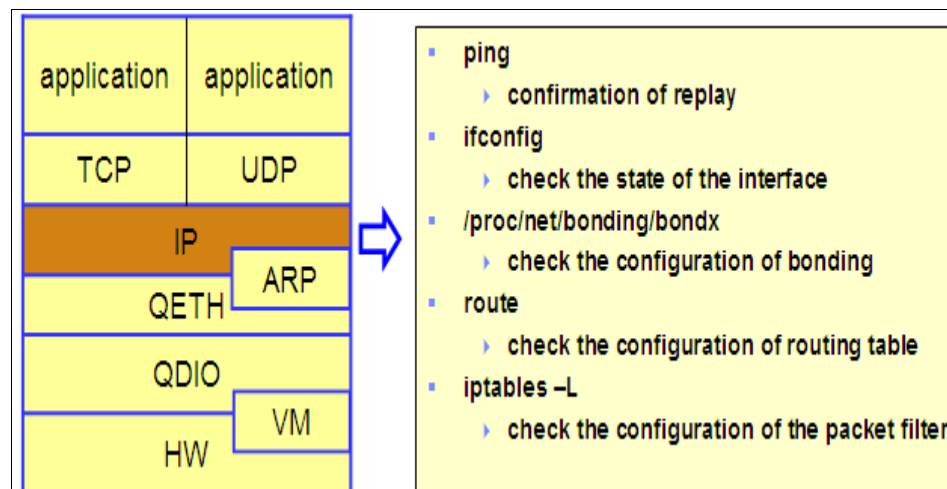


Figure 4-16 Survey in the IP layer

Figure 4-16 on page 126 summarizes the common tasks one could perform to survey in the IP layer.

```
# ping -c 3 -R 172.16.3.189
PING 172.16.3.189 (172.16.3.189) 56(124) bytes of data.
64 bytes from 172.16.3.189: icmp_seq=1 ttl=64 time=17.9 ms
RR: 172.16.3.131
    172.16.3.189
    172.16.3.189
    172.16.3.131

64 bytes from 172.16.3.189: icmp_seq=2 ttl=64 time=0.683 ms (same route)
64 bytes from 172.16.3.189: icmp_seq=3 ttl=64 time=0.635 ms (same route)

--- 172.16.3.189 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 201ms
rtt min/avg/max/mdev = 0.635/6.418/17.936/8.144 ms
```

option
-c: Determines the number of echo requests to send.
-R: Record route

← confirm the reply

Figure 4-17 Using ping for diagnosis

Figure 4-17 on page 127 show a sample ping output. The ping command tells 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 4-18 on page 127 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
(6)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 which are active (i.e UP) is displayed. The `-a` command-line parameter will force `ifconfig` to display all the interfaces which are configured irrespective of their state. Figure 4-19 on page 128 shows a sample `ifconfig` command output when executed with the `-a` command-line parameter.

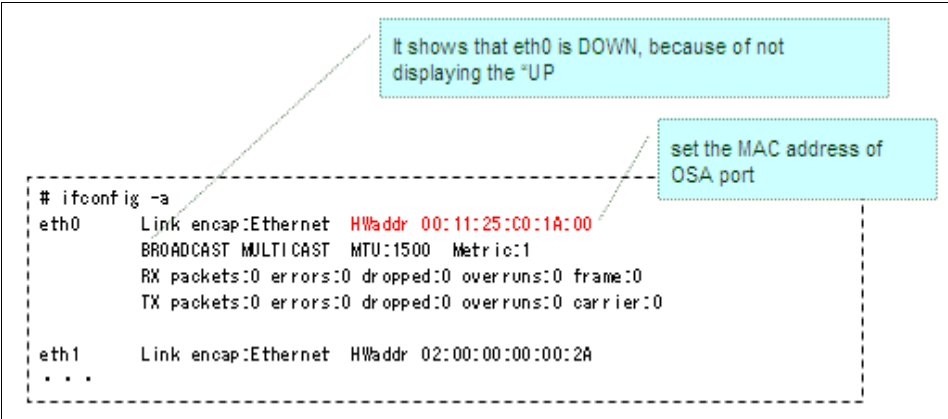


Figure 4-19 `ifconfig` command output indicating interface DOWN state

If the interface is down, then the troubleshooting should continue below the IP layer. If the IP address, subnet mask, etc 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 on page 128 shows a sample `route -n` output.

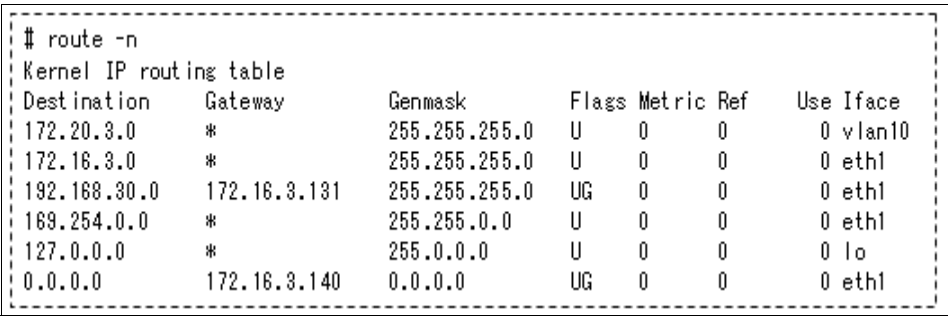


Figure 4-20 Checking the routing table

Sometime 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 on page 129 shows a sample iptables -L output which indicates that SSH connections coming from any host should be dropped.

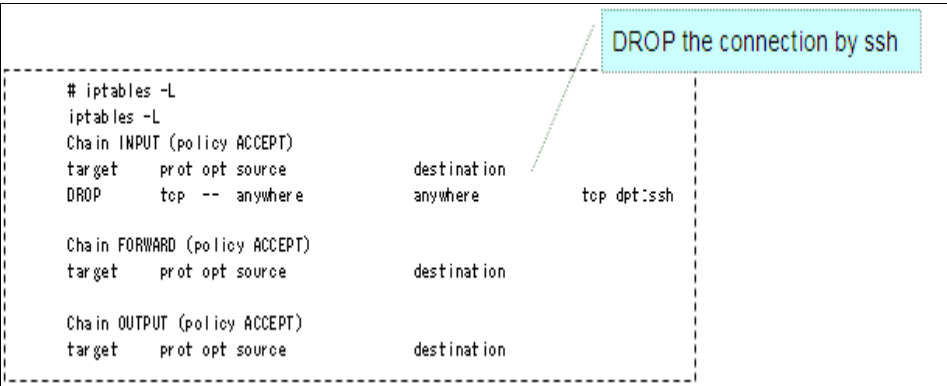


Figure 4-21 Checking the packet filtering configuration by using iptables

Survey in the ARP layer

Figure 4-22 on page 129 summarizes the ARP layer survey commands.

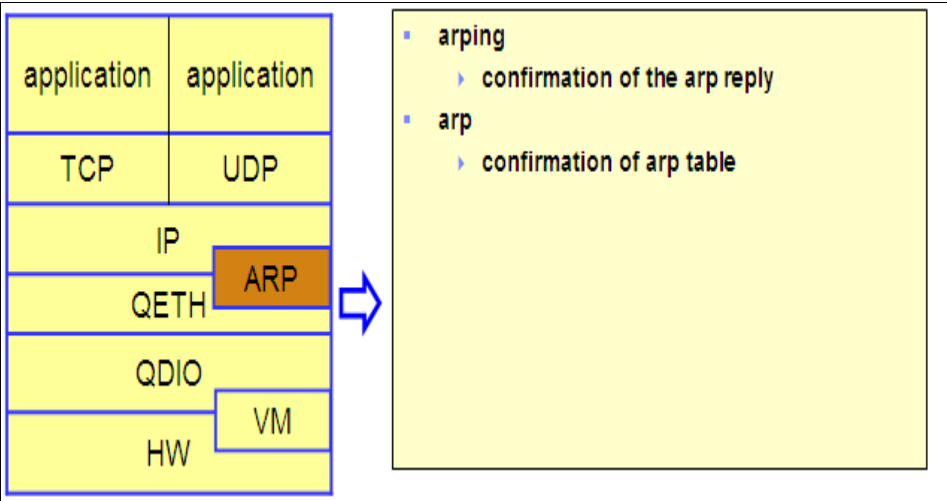


Figure 4-22 Survey in the ARP layer

arping command can be used to send ARP requestARP requests to a host. Figure 4-23 on page 130 shows a Sample outputsample output of an arping command execution. Such an ARP request updates the ARP cacheARP 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, probably. the troubleshooting should continue below the ARP layer. Figure 4-24 on page 130 shows how to check the ARP table using the arp command.

```
# arp
Address (1)           HWtype HWaddress (2)      Flags Mask (3)      Iface (4)
172.20.3.182          ether  00:03:47:98:77:6A  C                    bond0
```

(1) IP address of the other side
(2) MAC address of the other side
(3) entry type
 C : normal entry, drop out after a certain period of time
 M : persistent entry
 P : entry for Proxy ARP
(4) interface name
 ARP table is managed per interface

Figure 4-24 Using the arp command for diagnosis

Survey in the QETH layer

Figure 4-25 on page 131 shows the different things which can be surveyed in the QETH layer. The main thing which needs to be checked in this layer is whether the qeth device driver has recognised the devices. Figure 4-26 on page 131 shows a Sample outputsample output of the /proc/qeth file indicating how to check whether the network device is recognised by the qeth device driver.

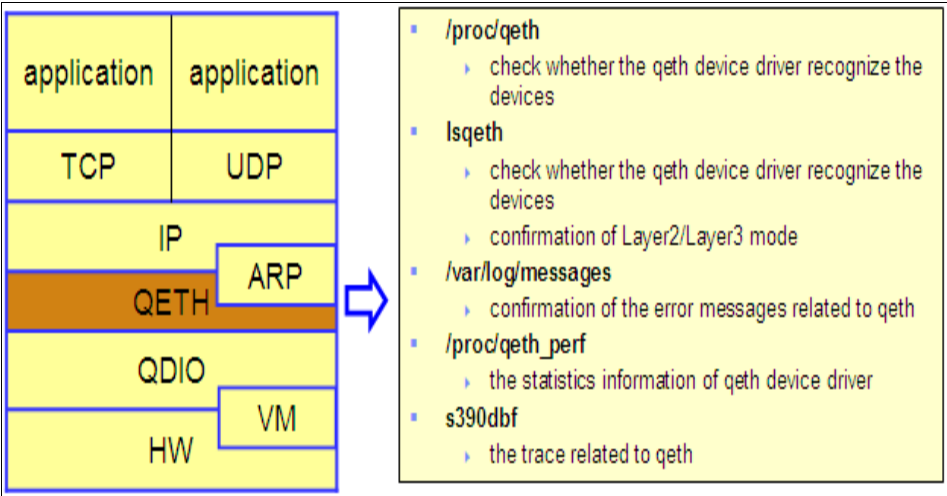


Figure 4-25 Survey in the QETH layer

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

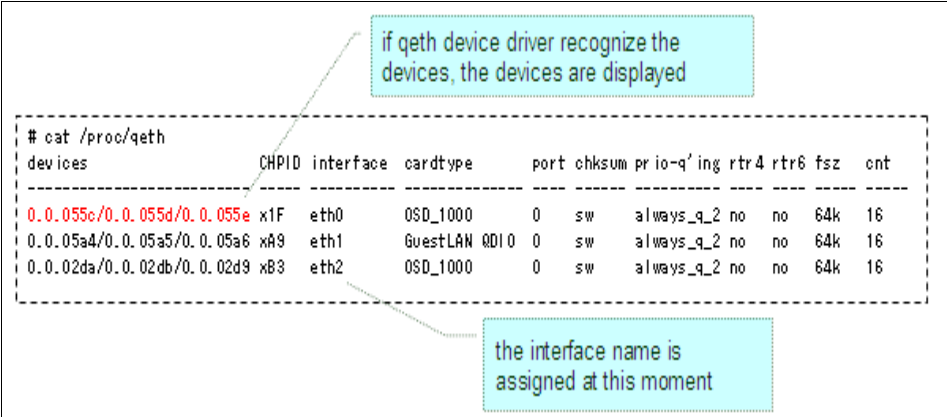


Figure 4-26 Using the `/proc/qeth` file information for diagnosis

Figure 4-27 on page 132 shows the output of the `lsqeth` command. 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.

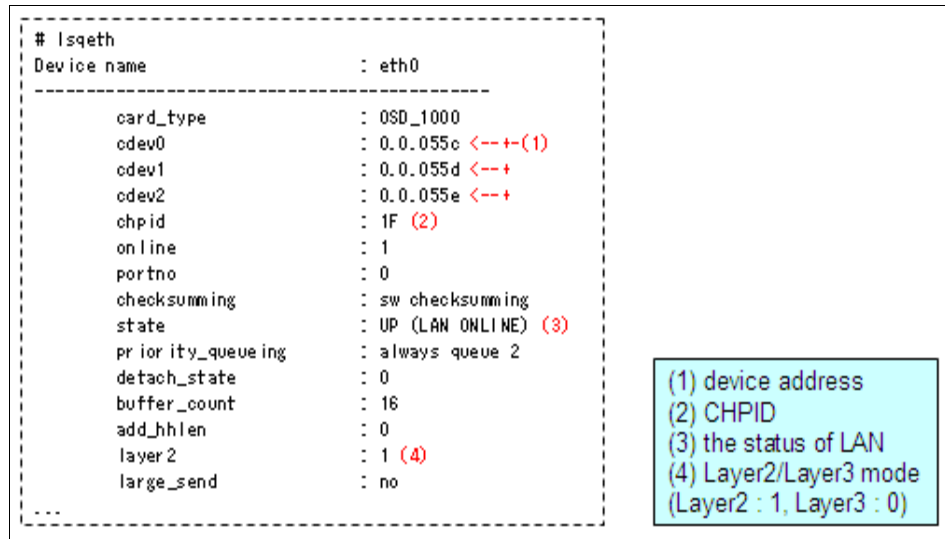


Figure 4-27 Using the `lsqeth` output for diagnosis

`/var/log/messages` file in Linux captures most of the common messages generated. This is a very important file referenced by system administrators, developers, etc to troubleshoot a problem. Figure 4-28 on page 132 shows sample messages logged on to the `/var/log/messages` file while there was a problem with one of the network device. The device name coming at the top of the sample output shown helps us to determine whether the problem is with the device we are working with, etc.

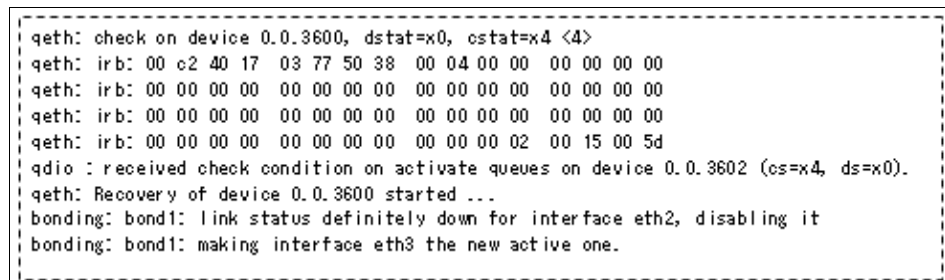


Figure 4-28 Using the `/var/log/messages` information for diagnosis

The number of packets transferred between the `qdio` and `qeth` drivers can be checked by examining the `/proc/qeth_perf` file. Figure 4-29 on page 133 shows a sample output with the number of packets transferred between the `qeth` and `qdio` indicated. Unexpectedly large packet counts will mean that there is a problem stproblem stst the `qeth` driver level or at the hardware level.

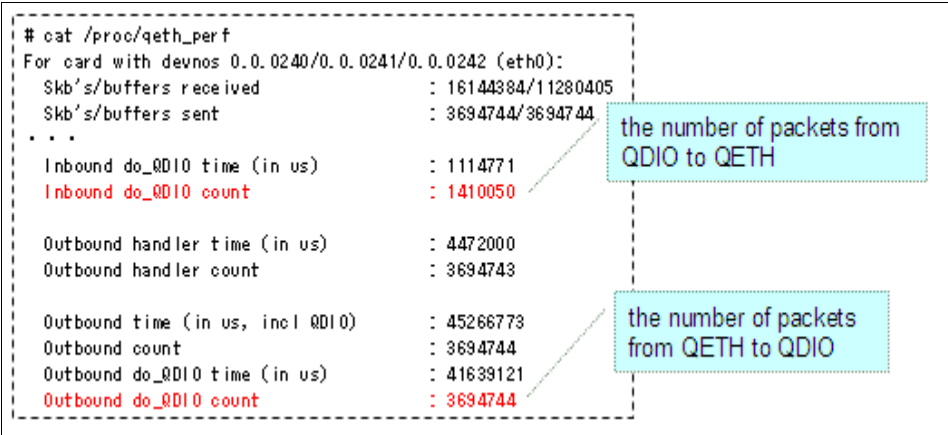


Figure 4-29 Using information from /proc/qeth_perf for diagnosis

Additional trace information for debugging can be found in the /proc/s390dbf/qeth* files. While reporting problem to IBM support, the content of these files needs to be provided for diagnosis. Figure 4-30 on page 133 shows the output of trace information:such file with trace information information for qeth.

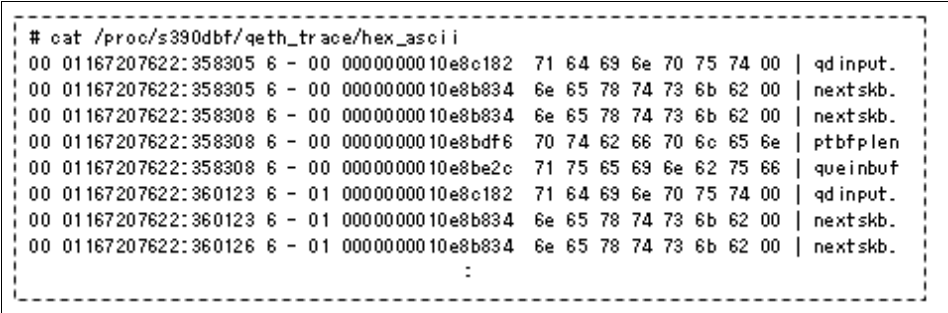


Figure 4-30 Using information from /proc/s390dbf/qeth_trace/hex_ascii for diagnosis

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 134 shows a sample output of the /proc/s390dbf/qdio_trace/hex_ascii.

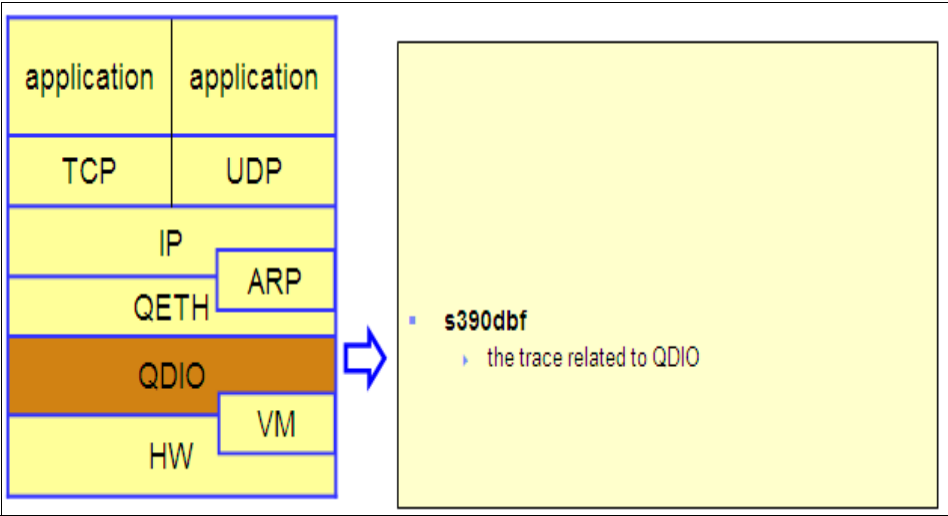


Figure 4-31 Survey in the QDIO layer

```
# 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 on page 135 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.

A sample output of the Q V NIC command is shown in Figure 4-34 on page 135. 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 the z/VM directory (USER DIRECT) to be checked.

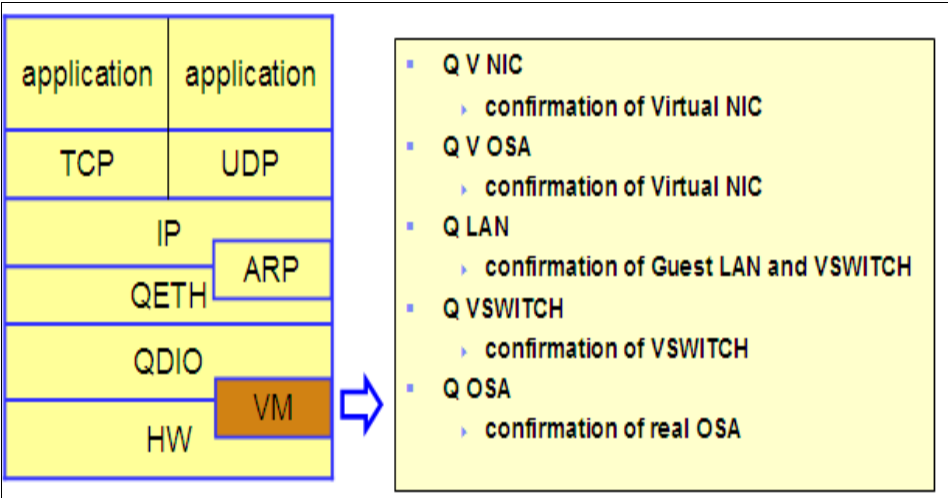


Figure 4-33 Survey in the VM layer

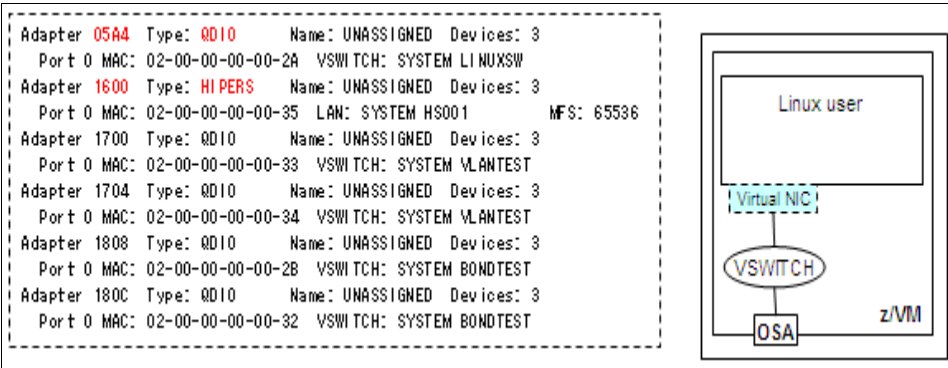


Figure 4-34 Using the Q V NIC output for diagnosis

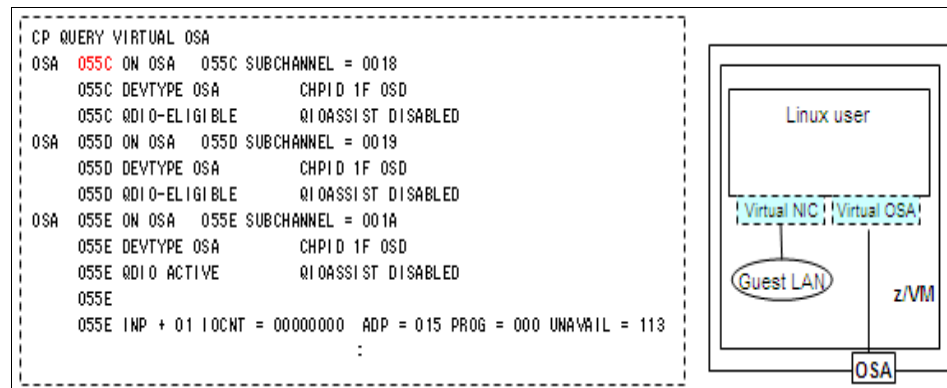


Figure 4-35 Using the Q V OSA output for diagnosis

Figure 4-35 on page 136 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 right OSA address is not displayed, then the USER DIRECT file needs to be checked.

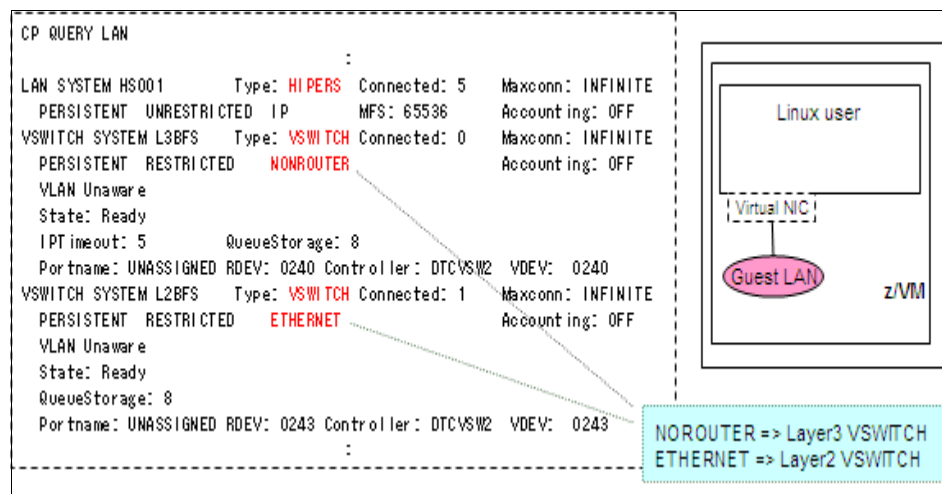


Figure 4-36 Using the CP Q LAN output for diagnosis

Figure 4-36 on page 136 shows the Q LAN output. This command output shows the information related to the HyperSockets, Guest LAN, VSWITCH, etc. If the LAN information is not displayed correctly, then IOCP needs to be checked in the case of HyperSockets and the z/VM SYSTEM CONFIG file for issues related to the Guest LAN and the VSWITCH.

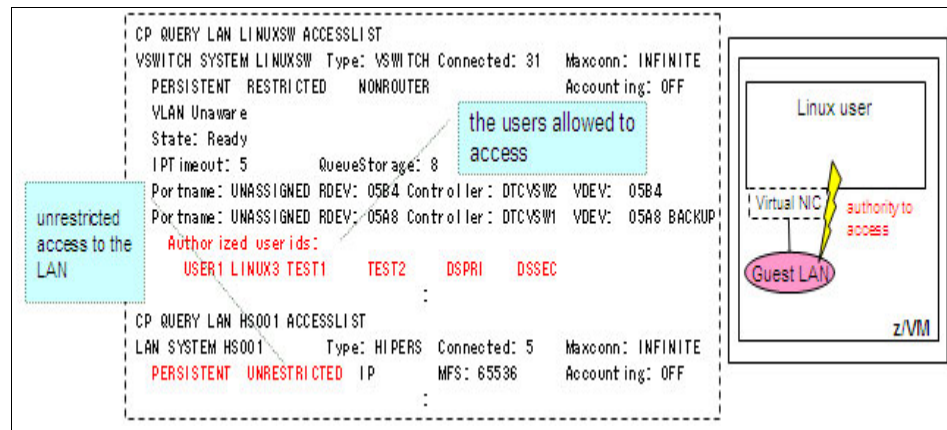


Figure 4-37 Using the Q LAN ACCESSLIST output for diagnosis

The Q LAN ACCESSLIST command output shown in Figure 4-37 on page 137 helps to show the authority to access the LAN. The authority is defined in the SYSTEM CONFIG file.

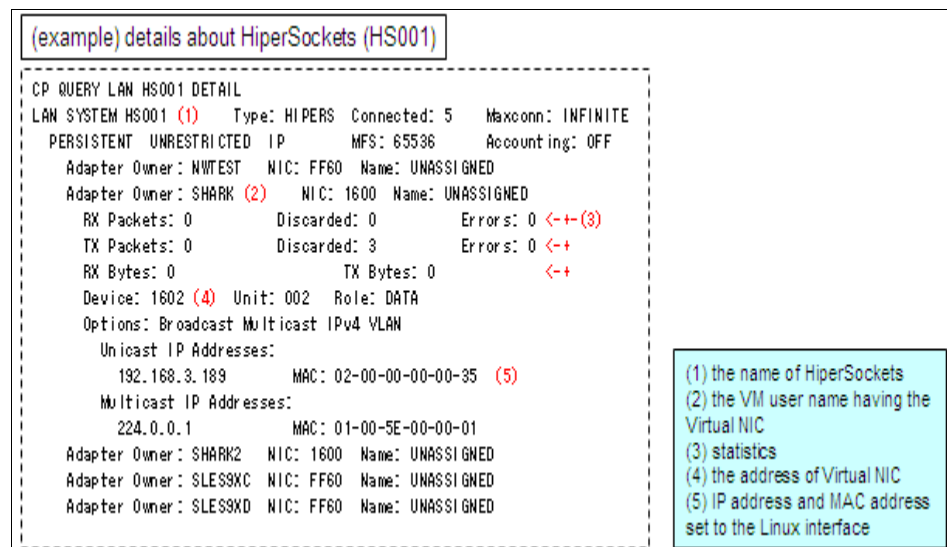


Figure 4-38 Using the CP Q LAN DETAIL output for diagnosis

The CP Q LAN DETAIL command output as shown in Figure 4-38 on page 137 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, etc. If the number of packets received and transmitted increases rapidly, then it may be indicating a problem.

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, etc.

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 OAT (OSA Address Table), the z/VM OSA/SF tool can be used. Figure 4-39 on page 138 shows a sample OSA/SF output. This output can be used to verify the MAC addresses and the IP addresses registered.

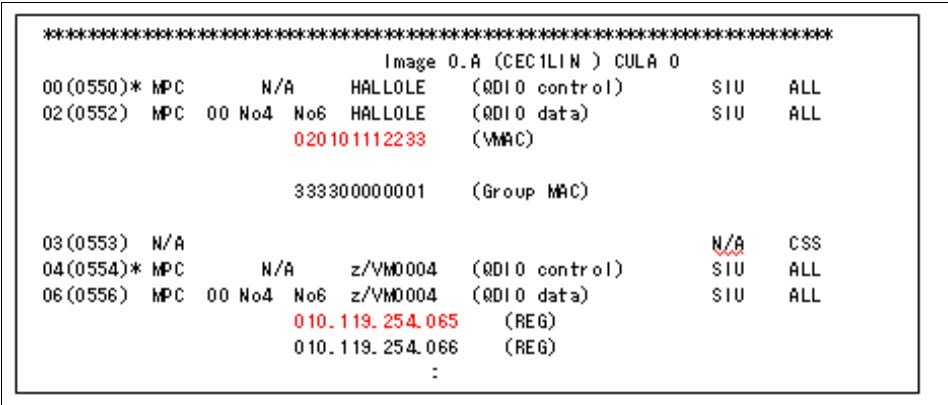


Figure 4-39 Using the z/VM OSA/SF tool for diagnosis

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

Survey in the hardware (HW) layer

Figure 4-40 on page 139 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 Linux kernel:parameter 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 the fact that 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 made online offline from Linux.

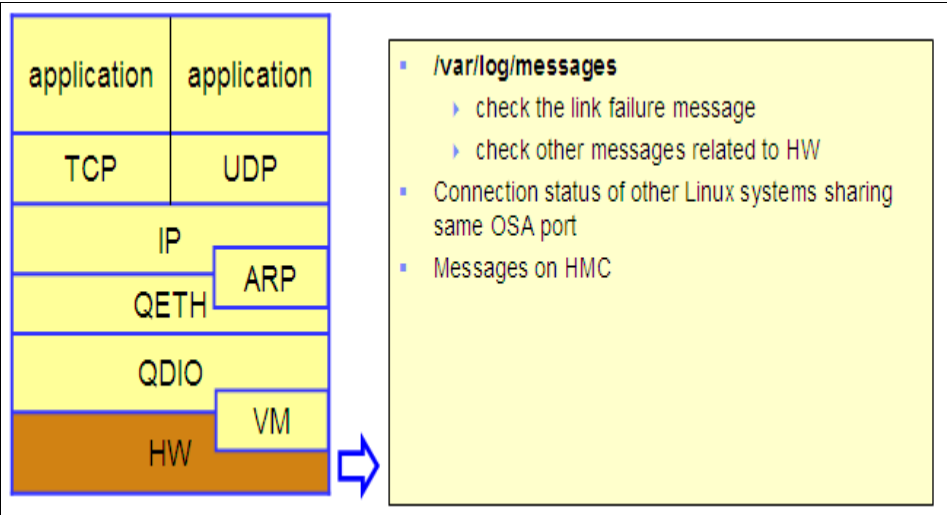


Figure 4-40 Survey in the hardware layer

cio_ignore can be setup permanently by adding an entry similar to the one shown in Example 4-2 on page 139 in to the /etc/zipl.conf file. It is also possible to setup the cio_ignore for a device dynamically as shown in Example 4-3 on page 139

Example 4-2 Sample entry to setup 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)
```

# 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

Figure 4-41 on page 140 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 are not correct, then, IOCP, z/VM USER DIRECT and `cio_ignore` to be checked. The `lscss` command does not show a device which is disabled (or hidden) using the `cio_ignore` setting.

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.

/var/log/messages file will contain information about the link status. Example 4-4 on page 140 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 needs to be checked, status of the other Linux systems sharing the same OSA port, if any, the hardware messages and retron codes displayed on the Hardware Management Console (Hardware Management Console (HMC)HMC).

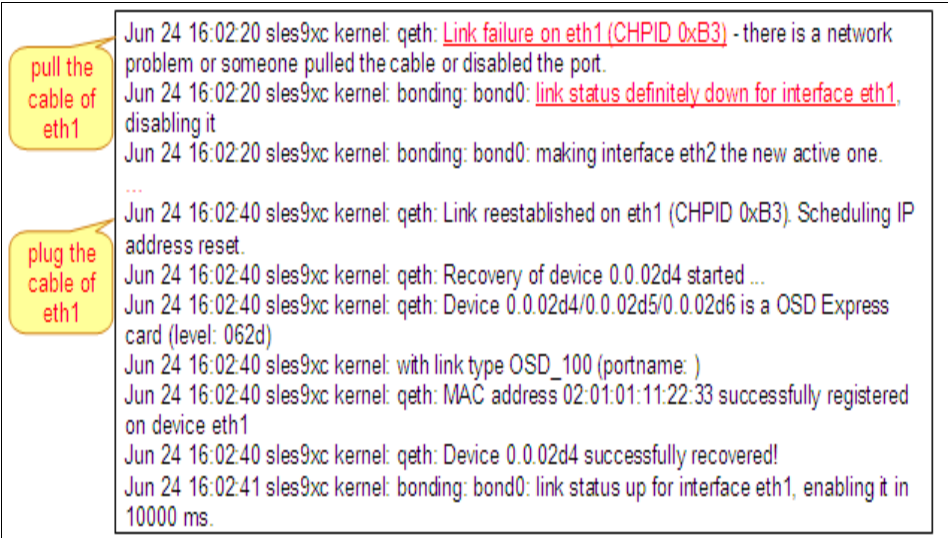


Figure 4-42 /var/log/messages output indicating link DOWN and UP status

Survey in the TCP/UDP layer

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

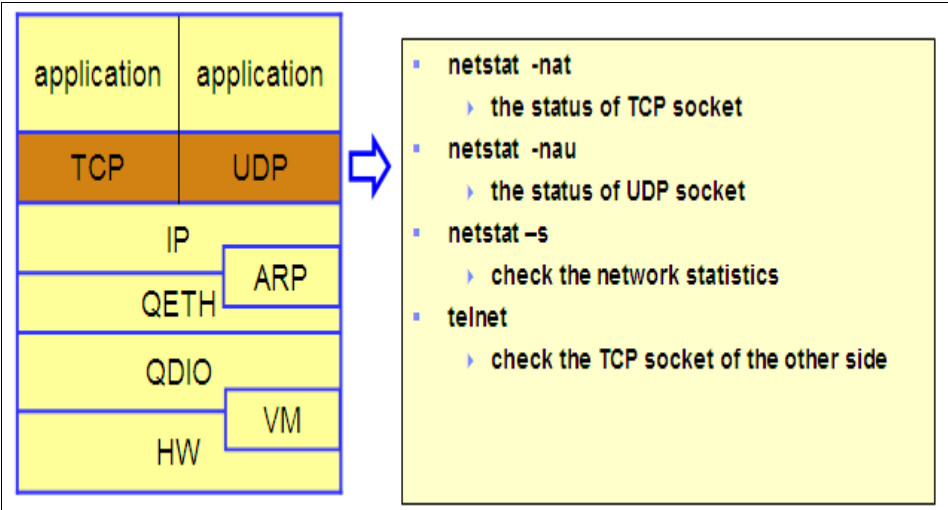
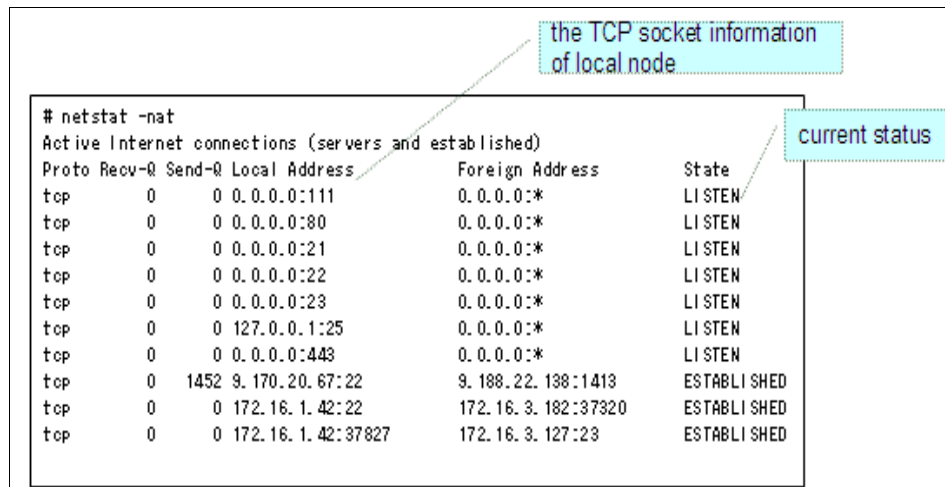
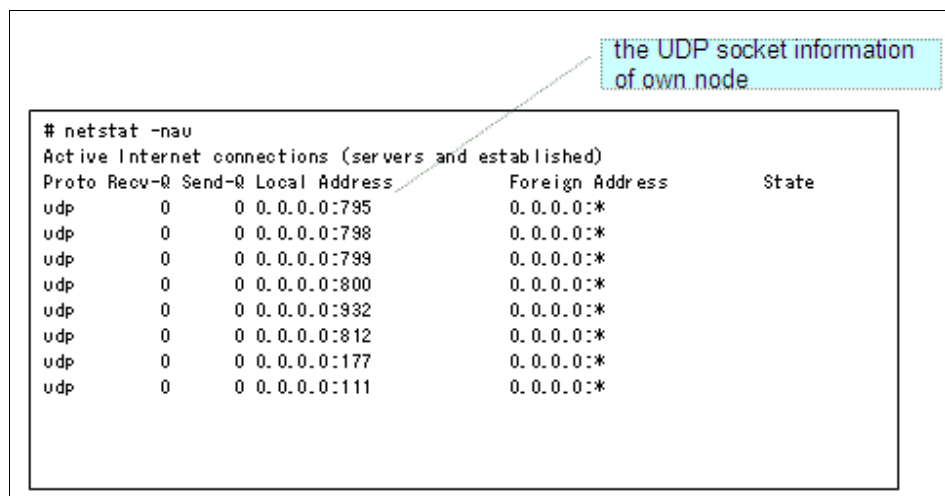


Figure 4-43 Survey in the TCP/UDP layer



```
# 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 0.0.0.0:1:25            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



```
# 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

Figure 4-44 on page 142 and Figure 4-45 on page 142 shows sample outputs for TCP (i.e. with the -t option) and for UDP (i.e. with the -u option) sockets respectively, using the netstat command. From this output, we can find what is the port on which the local machine is communicating and what is the destination port, etc. 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, etc.

Note: The column labelled as “State” will be blank for the netstat -tau output and the reason is that UDP is a stateless protocol.

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
...
```

Another useful netstat output is shown in Example 4-5 on page 143. If the count for the segments retransmitted is increasing, then usually, it is a point of worry.

A very quick way to check whether a system is listening on a TCP port TCP port is by doing a telnet to that system on to that port. This method cannot be used for UDP since it is a statless protocol. Figure 4-46 on page 144 shows an example of using the telnet command to establish a connection to TCP port 111. The result of a successful connection and an unsuccessful connection is displayed.

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.

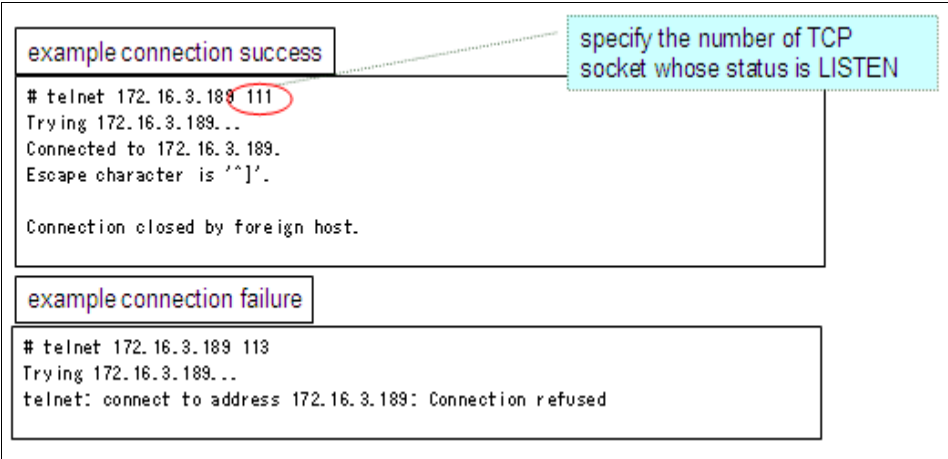


Figure 4-46 Using telnet to diagnose TCP socket status

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, etc.

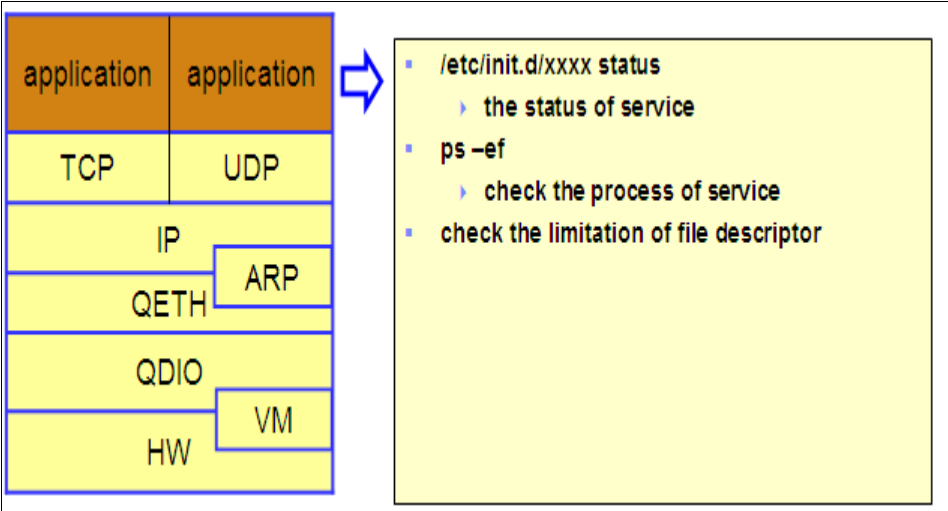


Figure 4-47 Survey in the application layer

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

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

Example 4-6 on page 145 shows a method to check the status of the sshd service. In this case, the output indicates that the SSH daemon is running.

Example 4-7 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
```

The sample output (last 3 lines) shown in Example 4-7 on page 145 indicate that the mail service (master and qmgr) is running and the SSH daemon is running with one user connected.

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. The file descriptor is needed when Linux uses a resource such as standard I/O, communication between processes, network socket besides the file on the physical disc, because these resources are treated virtually as file. Example 4-8 on page 145 shows the number of file descriptors opened by a process with process id of 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 client is limited by the file descriptor limitation. So, when new clients are not able to connect to an already running http daemon, for example, the reason could be that the http daemon reached the limit where it can no more 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 external to the local system. Again, here too, it is necessary to ask some logical questions to be able to proceed. Communication problem with a specific system only? Communication problem with all the other systems? Was 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 on page 146

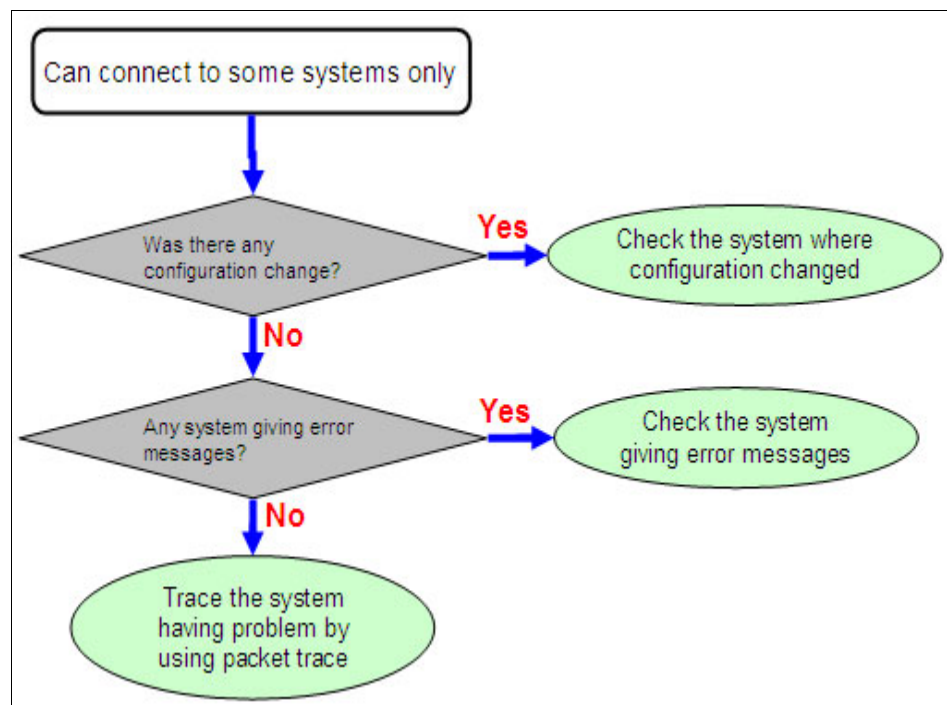


Figure 4-48 Problem determination outside the local machine

Packet tracing

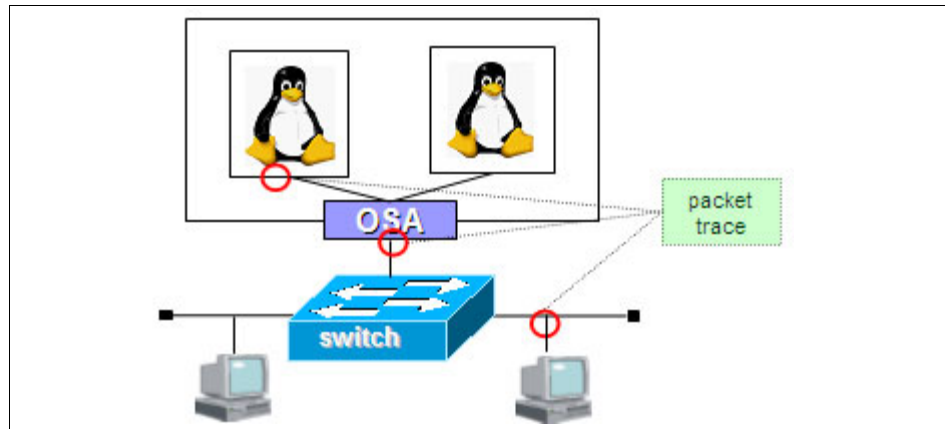


Figure 4-49 Using Problem determination:packet tracepacket trace for problem determination

Packet tracing 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 on page 147 shows a sample output of the Linux tcpdump tool. The tcpdump tool is used to capture the network packets on a system.

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 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 on page 148. As shown in the figure, 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.

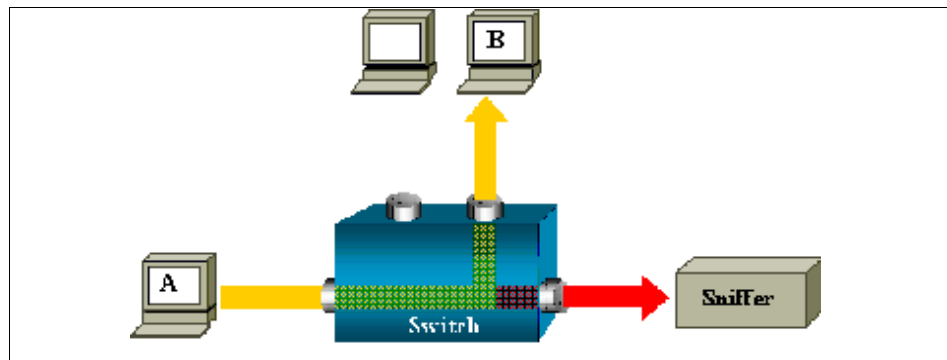


Figure 4-50 CISCO SPAN function

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 above). The CP

TRSOURCE command can be used to trace and record the data transfer. To analyze the output, the IPFORMAT utilityIPFORMAT utility can be used. To use the z/VM Guest LAN trace function, class C or higher privilege is required.

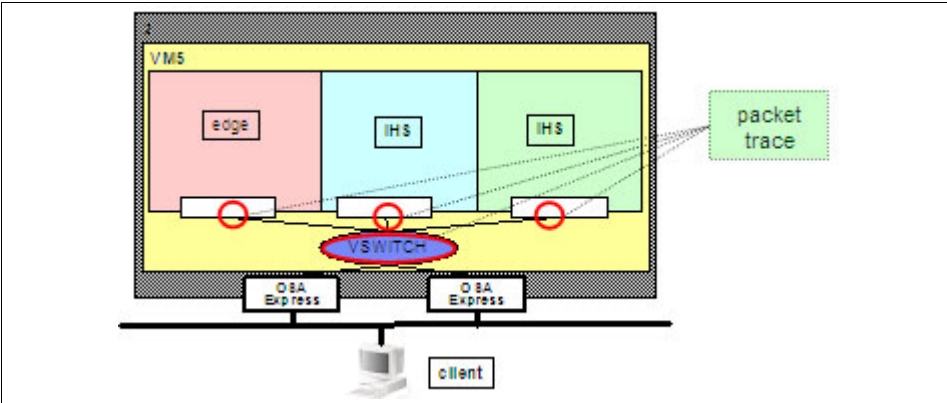


Figure 4-51 VM packet trace scenario

Figure 4-51 on page 149 shows a z/VM packet trace scenario. Assume that we need to capture the trace information between the “edge” system and the “IHS” system.

Step 1

Ready, T=0.01/0.01 13:12:19
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						

Step 2

Ready, T=0.01/0.01 13:13:30
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						

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

First we need to define the trace 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 the communication is attempted for 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 analyzing the same using the IPFORMAT utility. Figure Appendix 4-51, “VM packet trace scenario” on page 149 shows the definition of the trace, activating the same and then starting the trace.

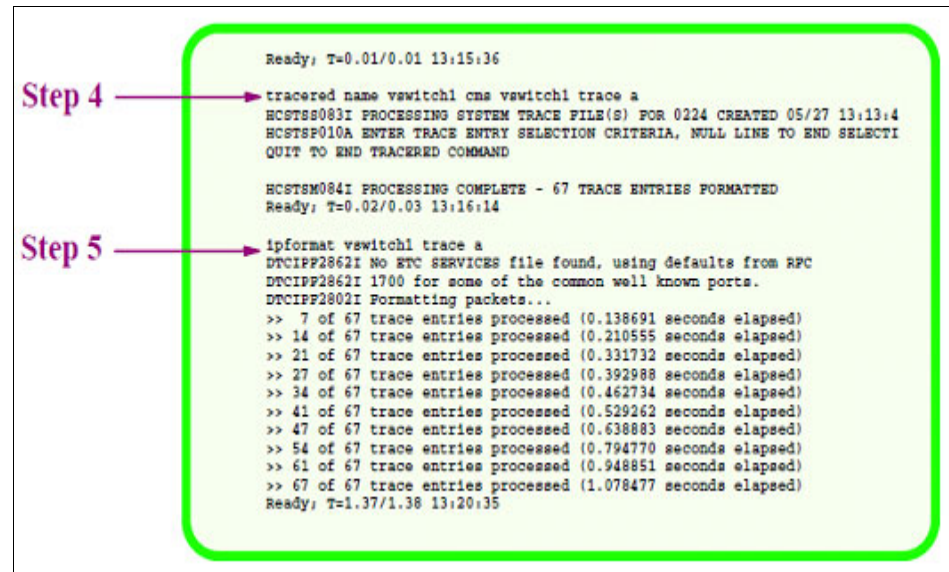


Figure 4-53 z/VM Guest LAN sniffing step 4-5

Figure 4-53 on page 150 illustrates stopping the trace and saving the trace record on to a CMS file. Also shown is the IPFORMAT command execution.

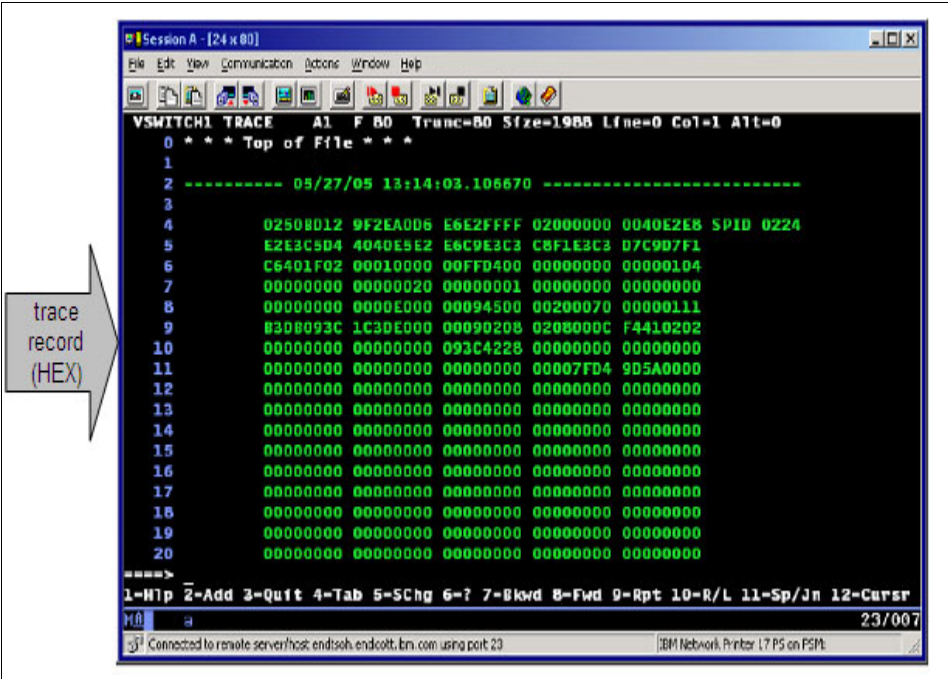


Figure 4-54 Sample trace record

Figure 4-54 on page 151 shows the trace record file contents. The contents are in the hexadecimal format. IPFORMAT utility processes this record and generates a more easy to read output. Figure 4-55 on page 152 shows the trace record after processing using the IPFORMAT utility.

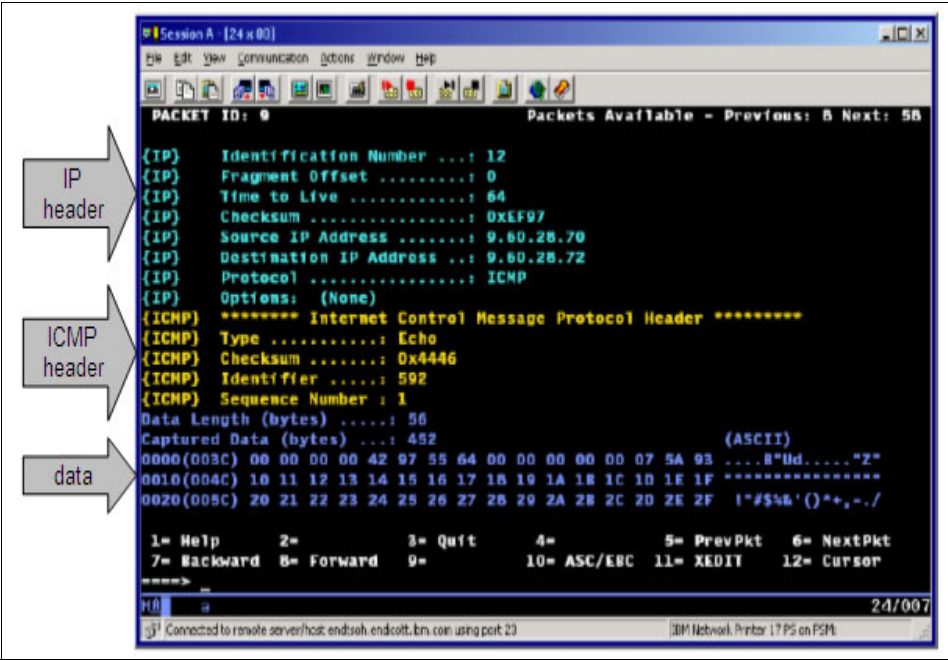


Figure 4-55 Sample IPFORMAT output

Example of a problem determination using packet trace

Figure 4-56 on page 152 shows Problem determination:example scenario
Problem determination:Example scenarioexample scenario for problem determination by using packet trace.

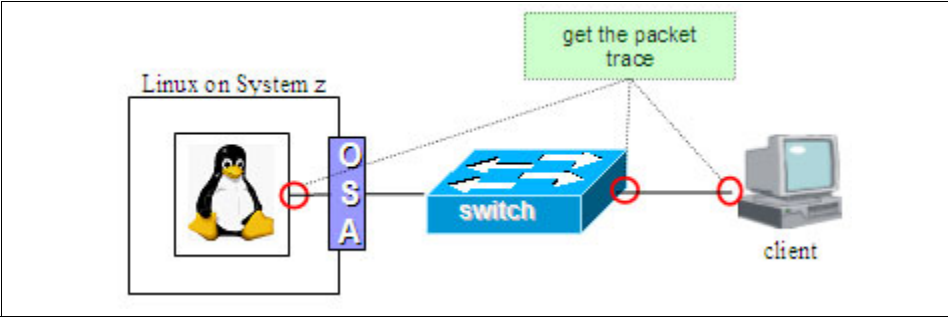


Figure 4-56 Example scenario for problem determination by using packet trace

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 gets frozen for a while then returns to normal state. Also, commands

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

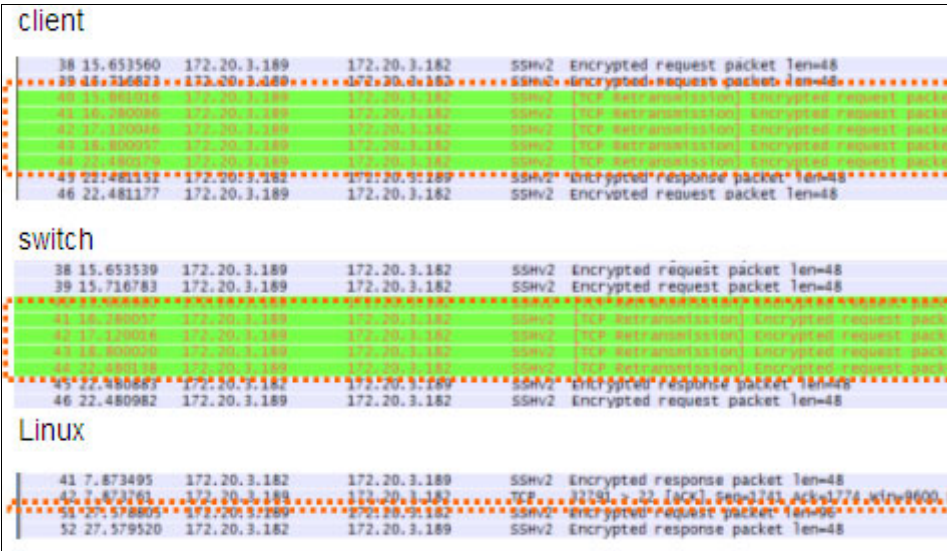


Figure 4-57 Example packet trace (switch port trouble)

As shown in Figure 4-57 on page 153, we can see that the client and the switch is 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. Probably, there is 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:settingOSA card port settings such as the negotiation mode, speed, etc.

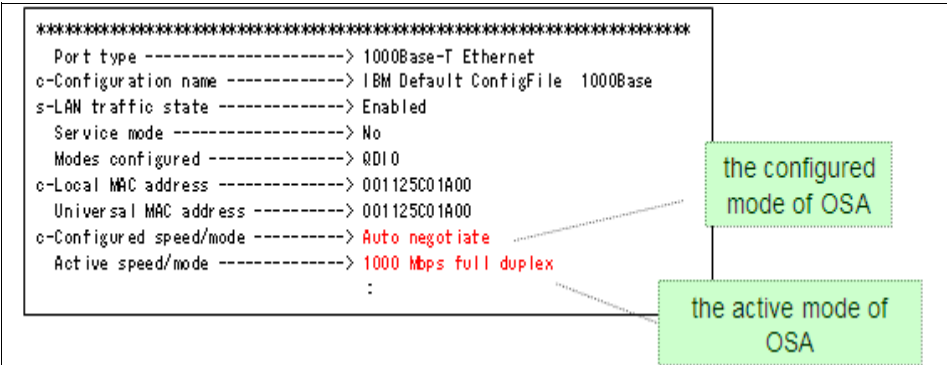


Figure 4-58 Sample OSA/SF output showing the OSA card port:settingOSA card port speed settings

This information can be compared with the switch side settings to ensure that both are configured and operating in the same mode, etc. As shown in Figure 4-58 on page 153, the OSA/SF tools shows the configured values for the OSA card OSA card and the active values.

4.4 Case studyCase studies

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

Case study 1 - eth0 interfaceeth0 interface down

In this case study we will discuss a scenario where the deactivation of eth0 interface causes the eth1 interface to stop function.

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, 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 on page 154 shows the scenario.

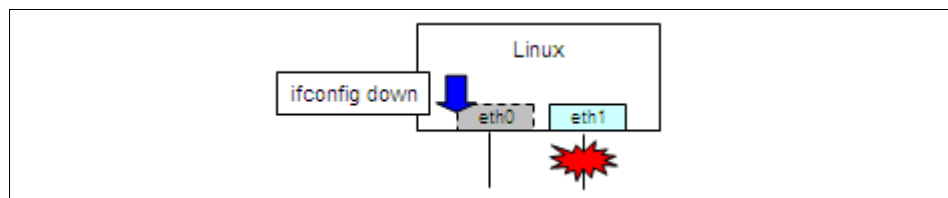


Figure 4-59 Case study example - eth0 interface down

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, it is identified 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 take-over

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

Problem

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

Data for analysis

- ▶ The status of devices 3700 and 3800
- ▶ ARP cache of z/VM TCPIP and the workstation executing the ping command
- ▶ Routing z/VM Linux guest:table information table 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 backup OSA device was 1000Mbps full duplex in spite of connecting to the 100Mbps 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 on page 156. 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

Not able to connect to the Linux system from the client using SSH.

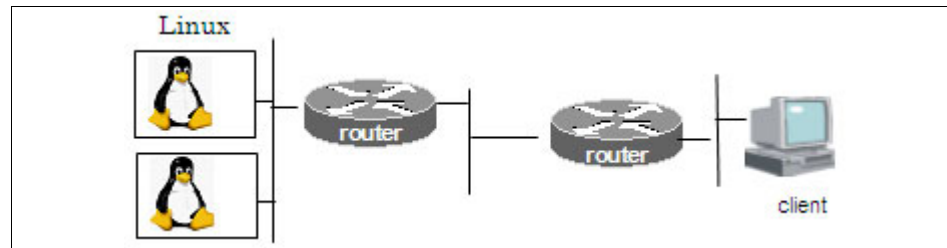


Figure 4-60 Case study example - Not able to connect using SSH

Data for analysis

- ▶ /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 and 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, Linux system:SSH server the SSH server on the Linux system is working correctly. Checking the router configuration, it was discovered that the MTU size MTU size setting of the router connected to the Linux system:network segment 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) DF (Do not Fragment) bit set. This causes the router connected to the Linux systems segment to send the ICMP Destination Unreachable (DU) DU Fragmentation Needed (FN) FN (ICMP Destination Unreachable / Fragmentation Needed) 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 router side. The tcpdump tcpdump was useful to troubleshoot this problem.

5



Performance Problem Determination

This chapter discusses performance problem determination for Linux on System z. The focus of this chapter is Linux running on z/VM.

In this chapter, the following topics are discussed:

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

5.1 What is a performance problem?

Typically, when the end 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 end user, this can vary. For example, when a system is running at 100% processor utilization and if the end 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.

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 (e.g.: applications or middlewares consume all the resources on the system), software problems (improper configurations, software bug, etc), 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 on page 159 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

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 on page 160, performance problem determination starts with understanding the problem and its occurrence.

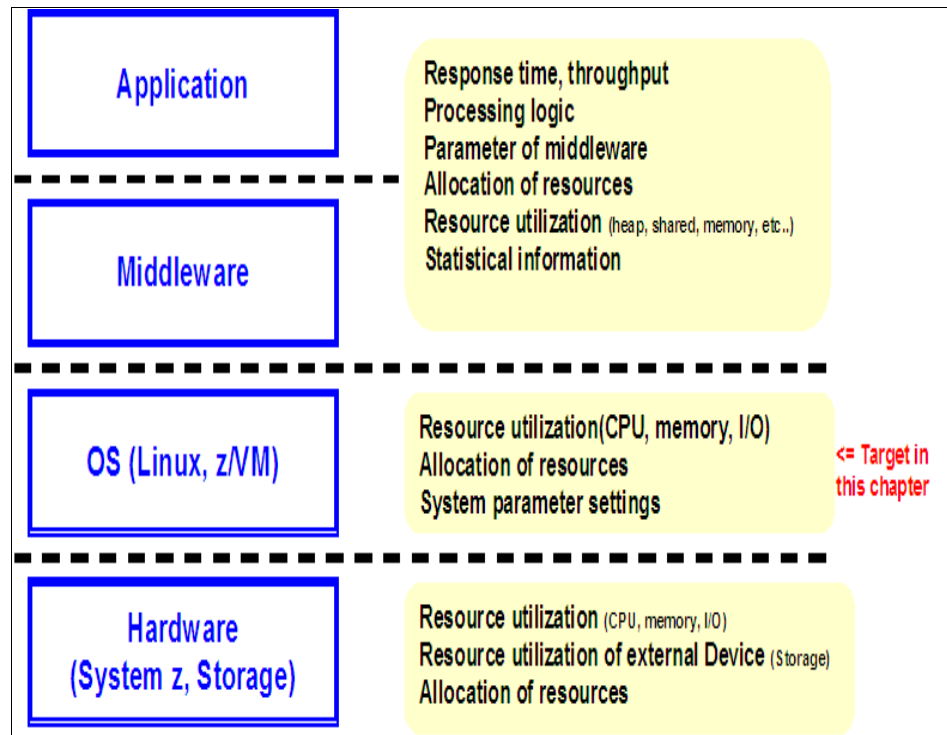


Figure 5-1 Performance problem determination layers

As shown in Figure 5-2 on page 160, the first step is to understand the problem clearly. As noted, it is important to understand the expectations of the end user of the system and what makes them 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, processing logic of the applications, etc. 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.

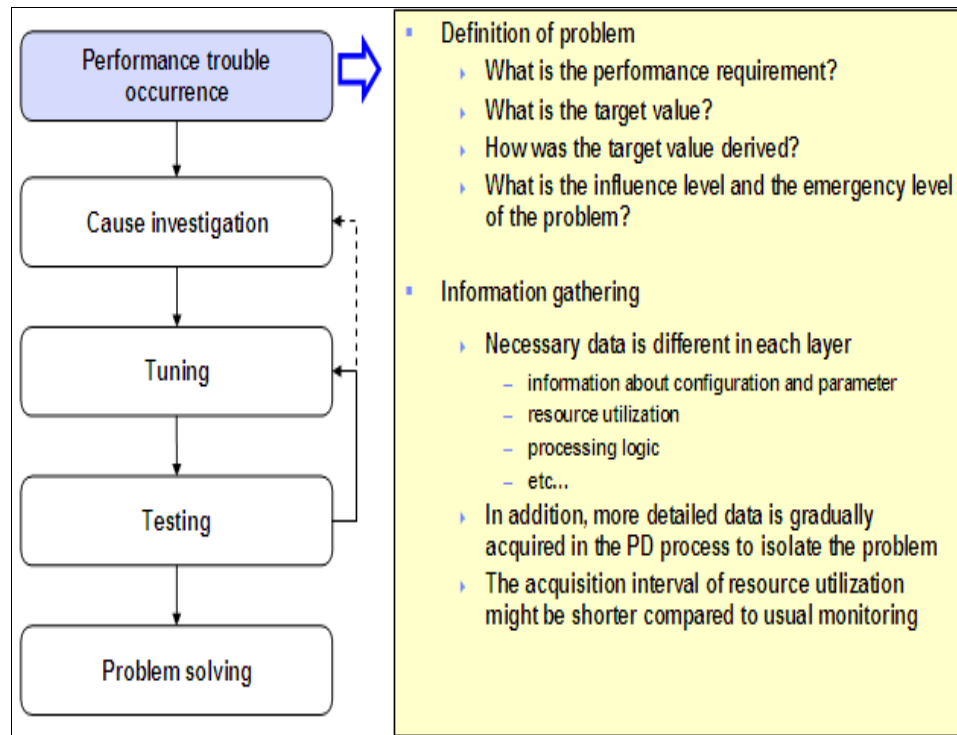


Figure 5-2 Performance problem determination - defining the problem

As indicated in Figure 5-3 on page 161, 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.

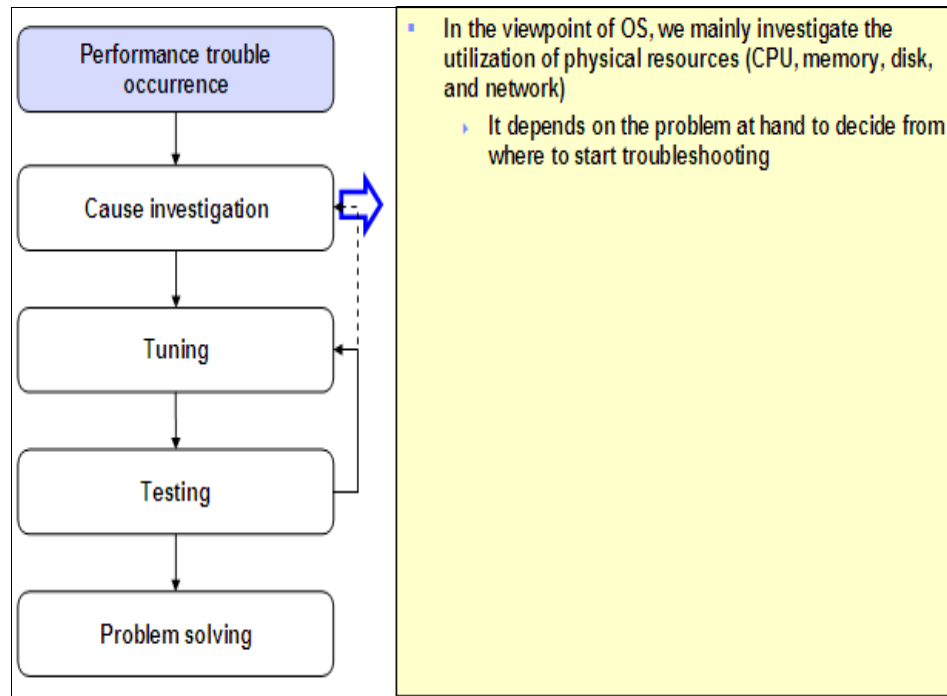


Figure 5-3 Performance problem determination - cause investigation

As shown in Figure 5-4 on page 162, the system is checked to find if 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 159. Sometimes it may happen that solving one bottleneck will introduce another bottleneck, which needs to be resolved.

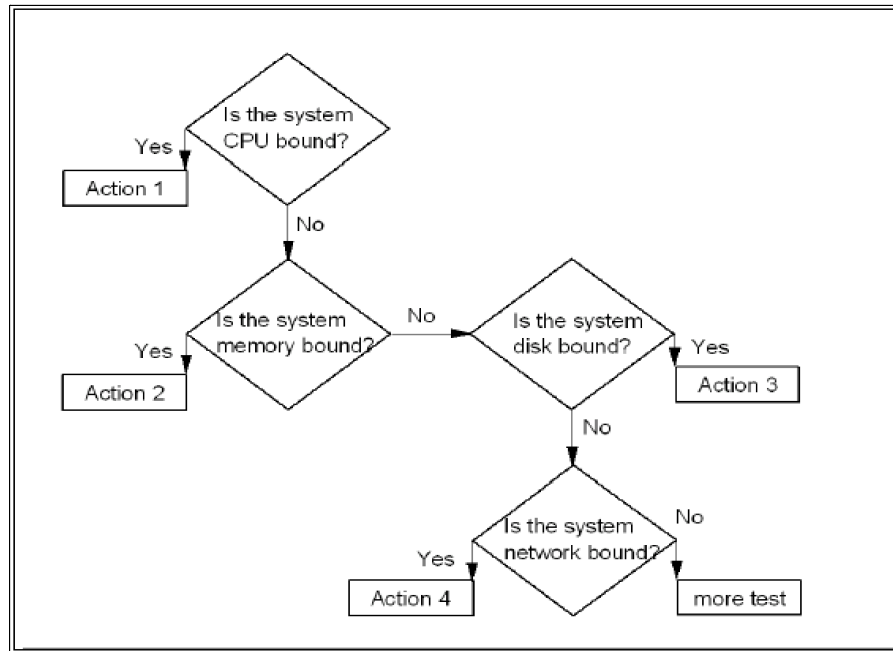


Figure 5-4 Performance problem determination - typical cause investigation flow

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, etc. When problems occur in an environment where every thing was working perfectly, the first thing to be checked is for any change (in the environment, on the system, etc). As shown in Figure 5-5 on page 163, 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. 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 sametime 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.

know is how much data is getting transferred, how many I/O operations are performed to transfer the data, etc. 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 57 for a summary of the common Linux tools used for information gathering.

5.2.2 CPU bottlenecks

In this section we will discuss about CPU bottlenecks and they ways to find the same. First let us take a brief look at how Linux and z/VM uses the CPU.

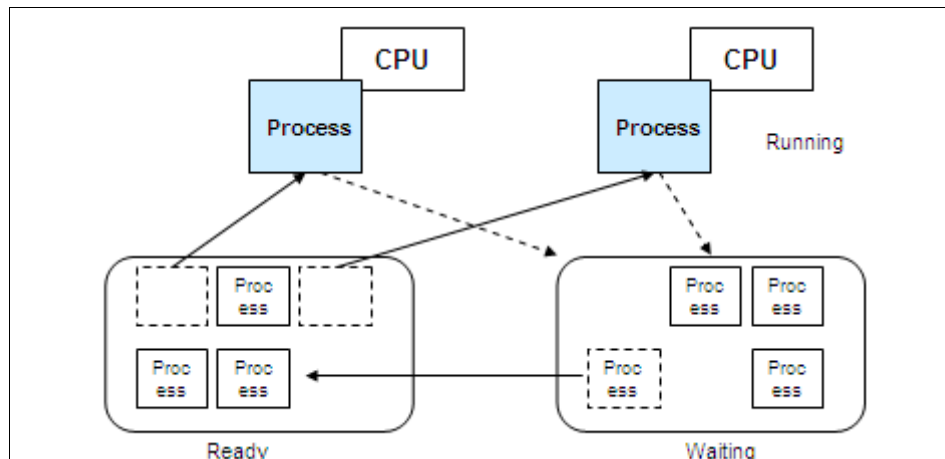


Figure 5-6 Linux CPU management

The Linux CPU management is illustrated in Figure 5-6 on page 164. Only one process gets executed on a given CPU at a time. The processes which 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 sometime, the Linux system moves out that process from the CPU and puts in another process. The processes waiting to enter the ready queue is maintained in a waiting queue.

Figure 5-7 on page 165 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.

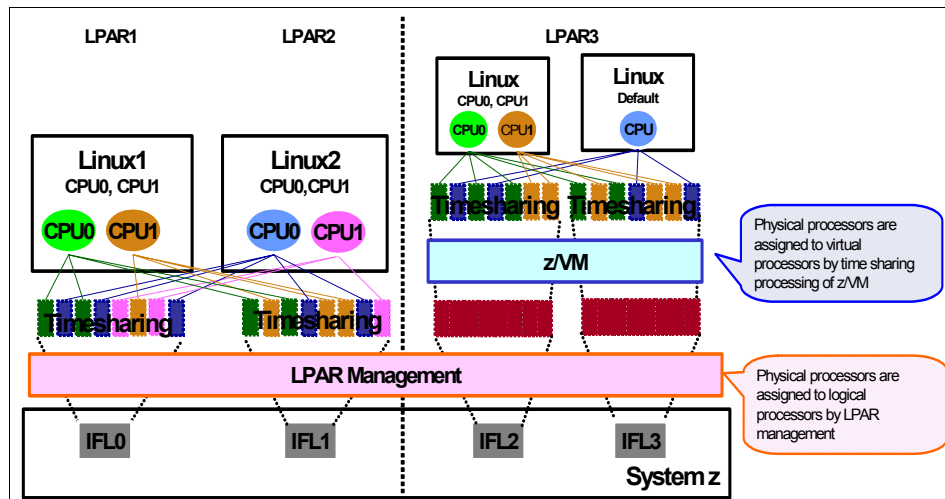


Figure 5-7 CPU sharing on System z

The CPU utilization is calculated as shown in Figure 5-8 on page 165. As can be seen from the 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.

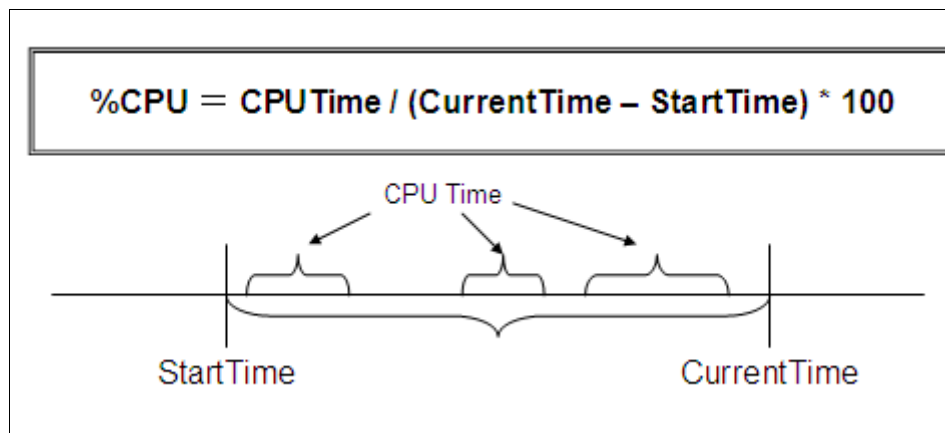


Figure 5-8 CPU utilization

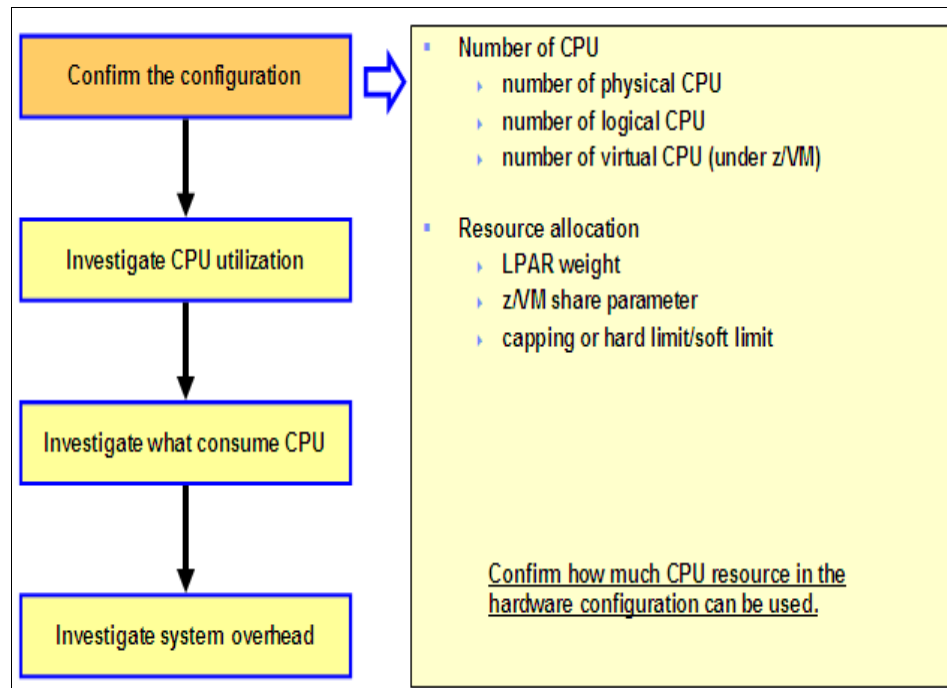


Figure 5-9 Flow of problem determination - Checking the configuration

As we discussed earlier, we start the problem determination by examining the system configuration. Figure 5-9 on page 166 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.

Once the configuration is verified, the next step is to investigate the cause for the high CPU utilization problem. Figure 5-10 on page 167 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 to be collected for the entire system, for each CPU and for each process. In case of z/VM, the same to be collected for each VM and also, the z/VM overhead to be measured. If the CPU utilization is 100%, then chances are there that the bottleneck is on the CPU. However, this is not always true since activities such as paging consumes lot of processing cycles. So, if the CPU utilization is reaching 100% mainly because of the paging operations, then we have a memory bottleneck.

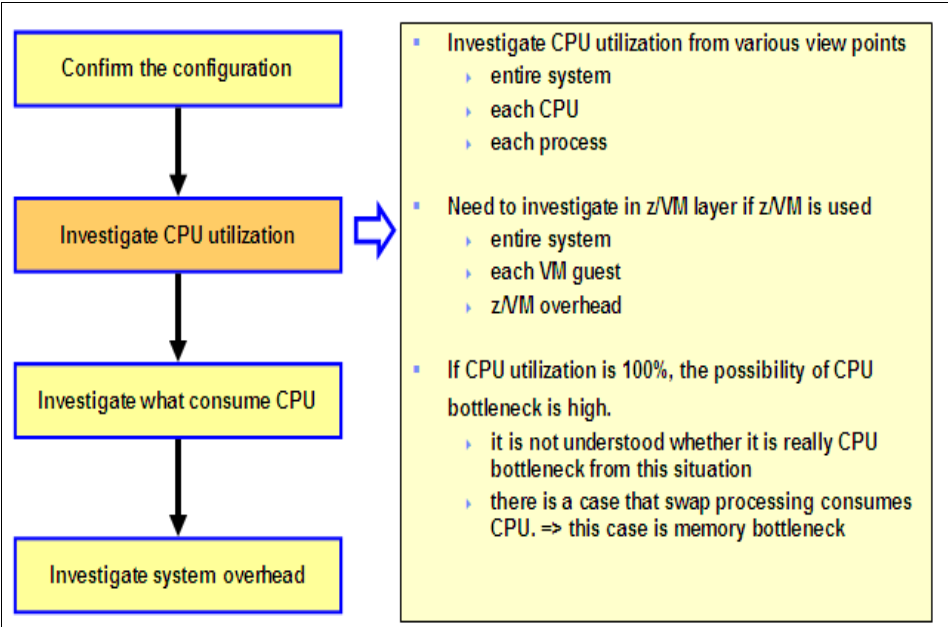


Figure 5-10 Flow of problem determination - investigating the CPU utilization

Figure 5-1 on page 167 summarizes the tools which can be used for acquiring the CPU utilization under Linux. This is not a complete list by any means. 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	

Name	Summary	Entire system	Each CPU	Each process
top	acquire CPU utilization of each process	X	X	X
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 utilizatuion 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 on page 168 shows a sample vmstat output where the CPU utilization of the entire system is available.

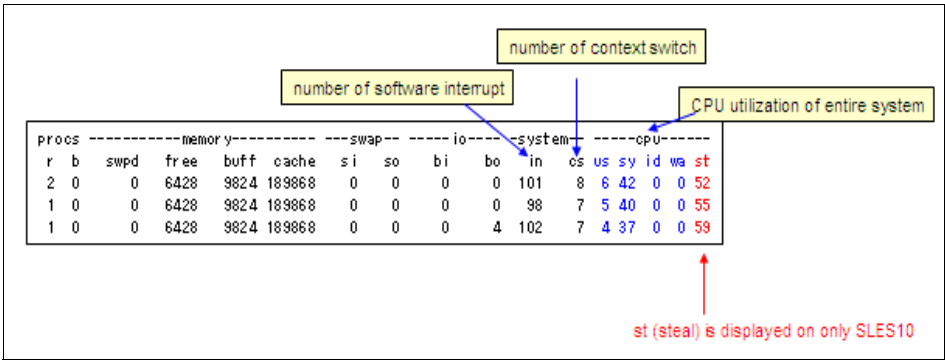


Figure 5-11 CPU utilization of entire system - vmstat

Figure 5-12 on page 169 shows a sample sar output where the entire CPU utilization as well as the utilization of each CPU is available.

sar -u -P ALL

15:05:57	CPU	%user	%nice	%system	%iowait	%idle
15:05:58	all	0.00	0.00	0.00	0.00	100.00
15:05:58	0	0.00	0.00	0.00	0.00	100.00
15:05:58	1	0.00	0.00	0.00	0.00	100.00

CPU utilization of entire system

CPU utilization of each CPU

Figure 5-12 CPU utilization of entire system - sar

Figure 5-13 on page 169 shows a case where from the vmstat output it looks like 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 on page 169 indicate 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.

output of vmstat

procs	-----memory-----	---swap---	io---	system--	cpu---
r b	swpd free buff cache	si so	bi bo	in cs us sy	id wa
1 0	0 3756940 24152 186376	0 0	0 52	210 42 6 44	50 0
1 0	0 3756940 24152 186376	0 0	0 0	203 31 7 43	50 0
1 0	0 3756940 24152 186376	0 0	0 0	203 29 6 44	50 0

it seems to be room because of idle=50%

output of sar

11:15:42	CPU	%user	%nice	%system	%iowait	%idle
11:15:43	all	4.50	0.00	45.50	0.00	50.00
11:15:43	0	0.00	0.00	0.00	0.00	100.00
11:15:43	1	9.00	0.00	91.00	0.00	0.00
11:15:44	all	6.00	0.00	44.00	0.00	50.00
11:15:44	0	0.00	0.00	0.00	0.00	100.00
11:15:44	1	12.00	0.00	88.00	0.00	0.00

It is CPU bottleneck because all of one CPU is used.

In general, Linux processes one process on one processor. In multi-processor environment, it is necessary to note that CPU utilization become the average of each CPU utilization.

output of mpstat

11:15:46	CPU	%user	%nice	%system	%iowait	%irq	%soft	%idle	intr/s
11:15:46	all	3.50	0.00	46.50	0.00	0.00	0.00	50.00	202.00
11:15:46	0	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00
11:15:46	1	7.00	0.00	93.00	0.00	0.00	0.00	0.00	0.00

Figure 5-13 Utilization of each CPU

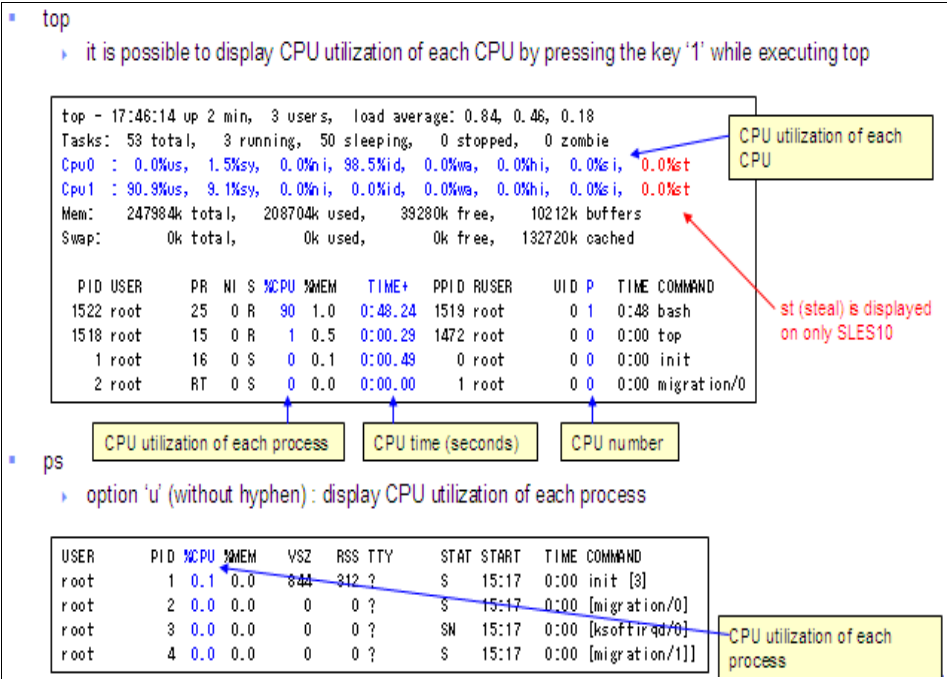


Figure 5-14 CPU utilization of each process

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 170 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 too.

Figure 5-15 on page 171 shows the z/VM Performance Toolkit output indicating the percentage utilization for each physical processor.

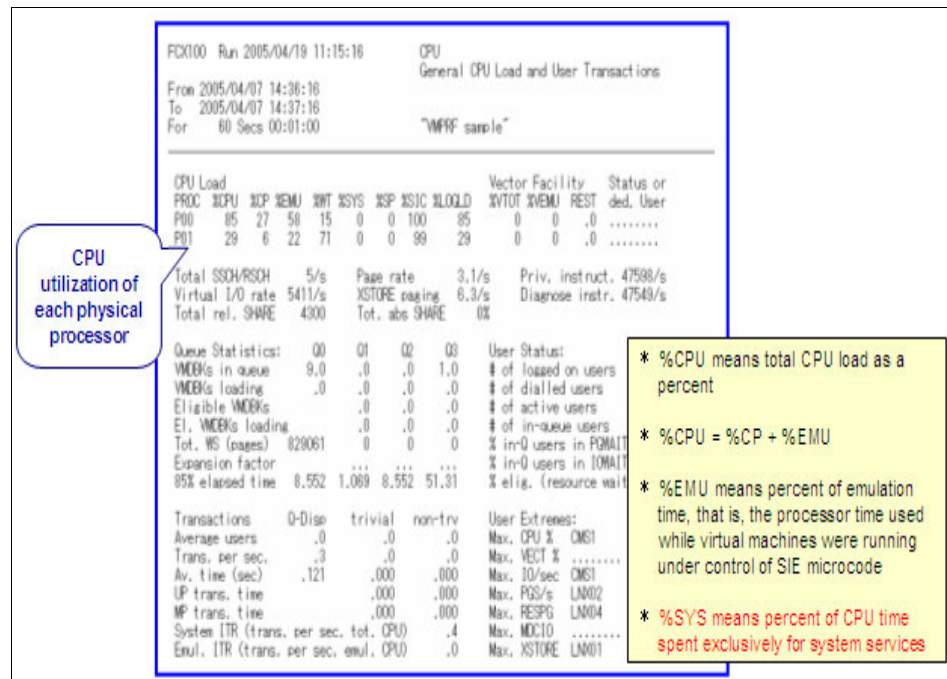


Figure 5-15 Individual processor utilization report using the z/VM Performance Toolkit

Depending on the Performance Toolkit report screen, the CPU utilization is displayed as average of all CPUs or total of all CPUs as follows.

- ▶ Processor Summary Log Screen (FCX239)
 - Pct Busy: displays average of all CPUs, and Max=100%
- ▶ CPU Load and Transactions screen (FCX100)
 - %CPU: displays CPU utilization of each physical CPU, and Max=100%
- ▶ User Resource Usage Screen (FCX112)
 - %CPU: displays total of all CPUs, and Max=100% x (number of physical CPU)
- ▶ General Performance Data by Time screen (FCX101)
 - CPU: display total of all CPUs, and Max=100% x (number of physical CPU)

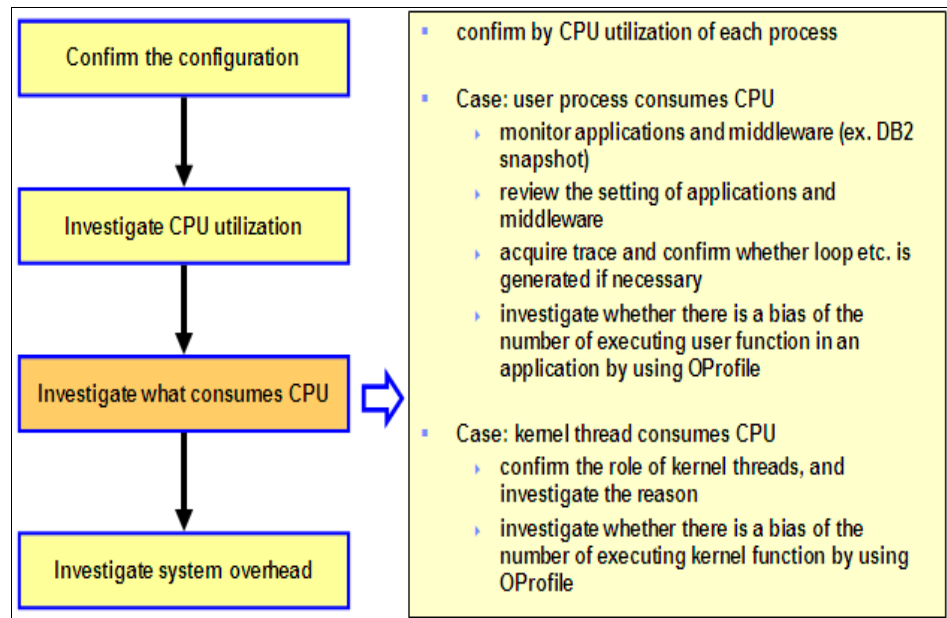


Figure 5-16 Flow of problem determination - investigate what consumes CPU

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 on page 172 summarizes the action points in each case.

kswapd is a Linux kernel thread which 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 which commits the transaction of filesystems using a journal such as the ext3 filesystem. High CPU utilization by kjournald indicate that a lot of CPU is being consumed for filesystem transaction commit to journal area due to lot of disk I/O.

pdflush is a kernel thread which writes dirty page cache to the disk. High CPU utilization by pdflush too indicate that there is lot of disk I/O since since it is actually writing dirty page caches to the disk very often.

When software IRQ load on the Linux system increaes, 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, then it means that the system is under heavy software interrupt load. This can happen because of very high disk I/O, heavy network traffic, etc.

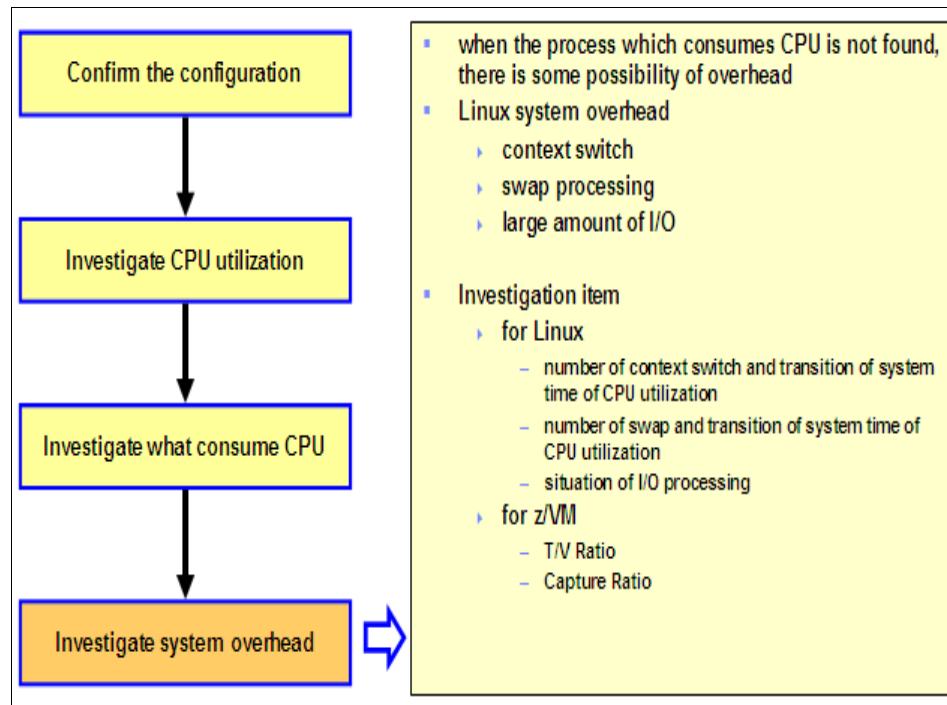


Figure 5-17 Problem determination flow - investigate system overhead

What happens if we do not find a specific process using lot of CPU but the individual CPU utilization is reported as high? Well, 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 which are considered as overhead are swap processing and large amount of I/O. As summarized in Figure 5-17 on page 173, the number of context switches, swapping, large I/O are the reasons behind increase in system overhead and hence high CPU utilization.

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, capture ratio is a measurement that shows whether the system can efficiently use the resource. Capture ratio is the ratio of user time to (system time + user time). This is illustrated in Figure 5-18 on page 174.

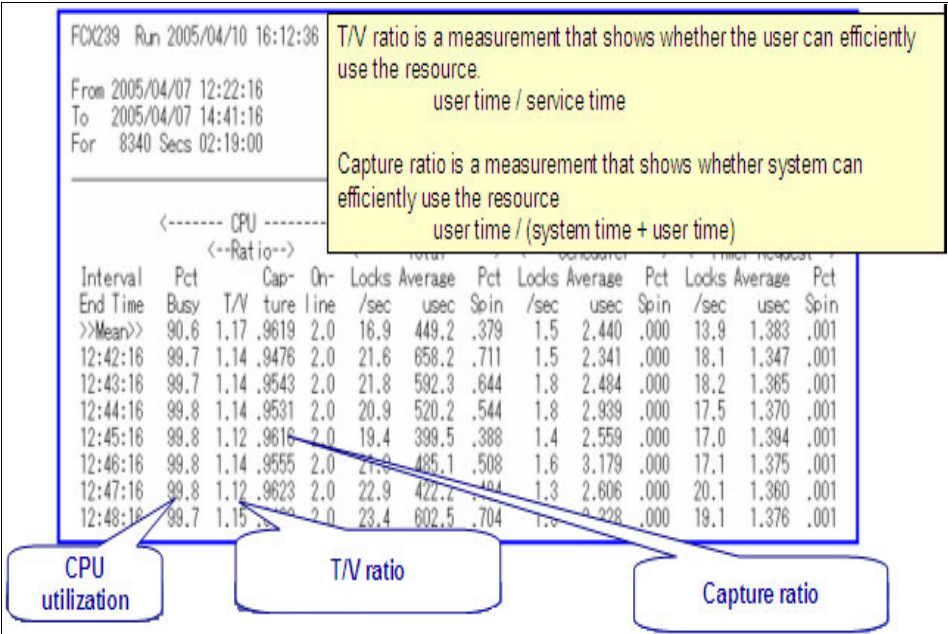


Figure 5-18 VM overhead check

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 as 4KB. This is illustrated in Figure 5-19 on page 175. The virtual memory is a logical layer between the application memory requests and the physical memory.

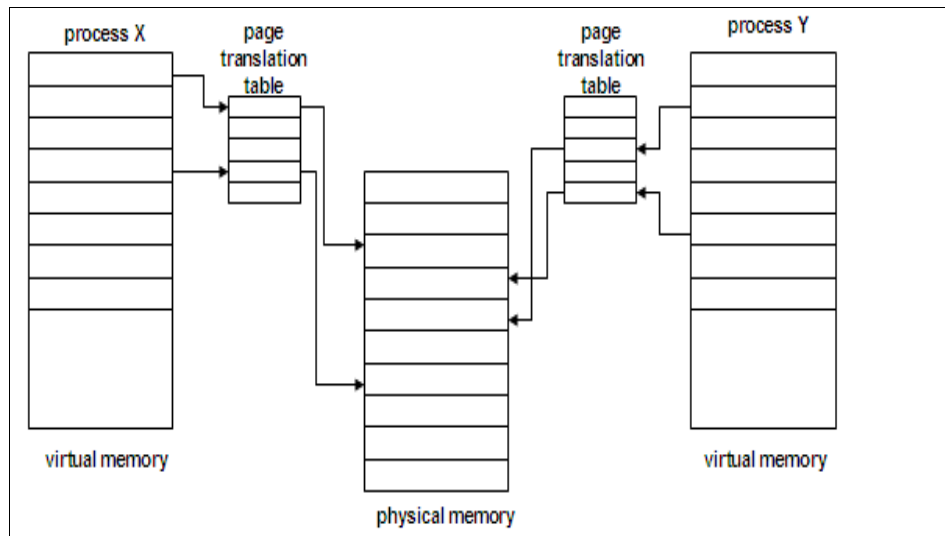


Figure 5-19 Linux memory management

When the main memory is not enough for the system to execute processes, the system will try to move some stuff from the main memory to the disk. This process is called as swapping or paging. Swapping occurs when the system needs more memory than what 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.

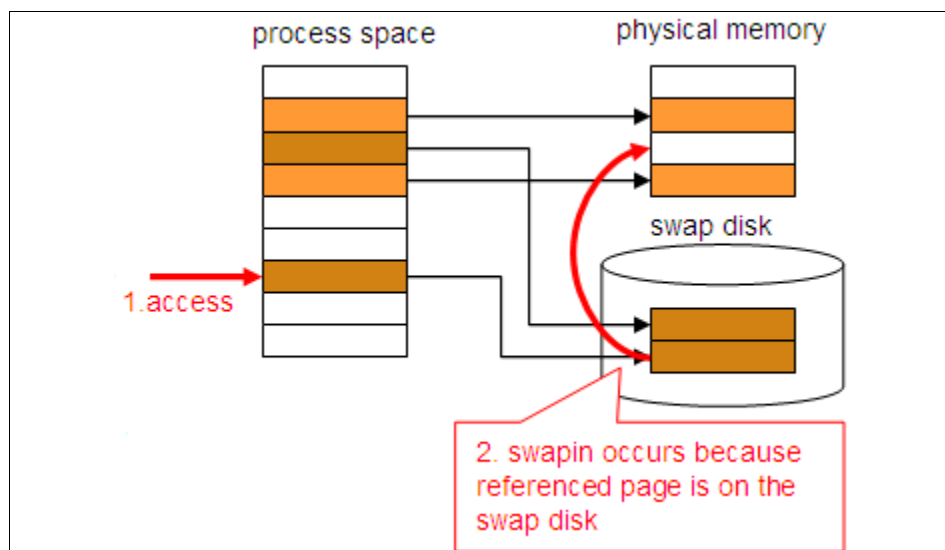


Figure 5-20 Swapping

When the swapped-out pages are to be accessed, they are swapped-in to the memory again. Figure 5-20 on page 175 illustrates swapping.

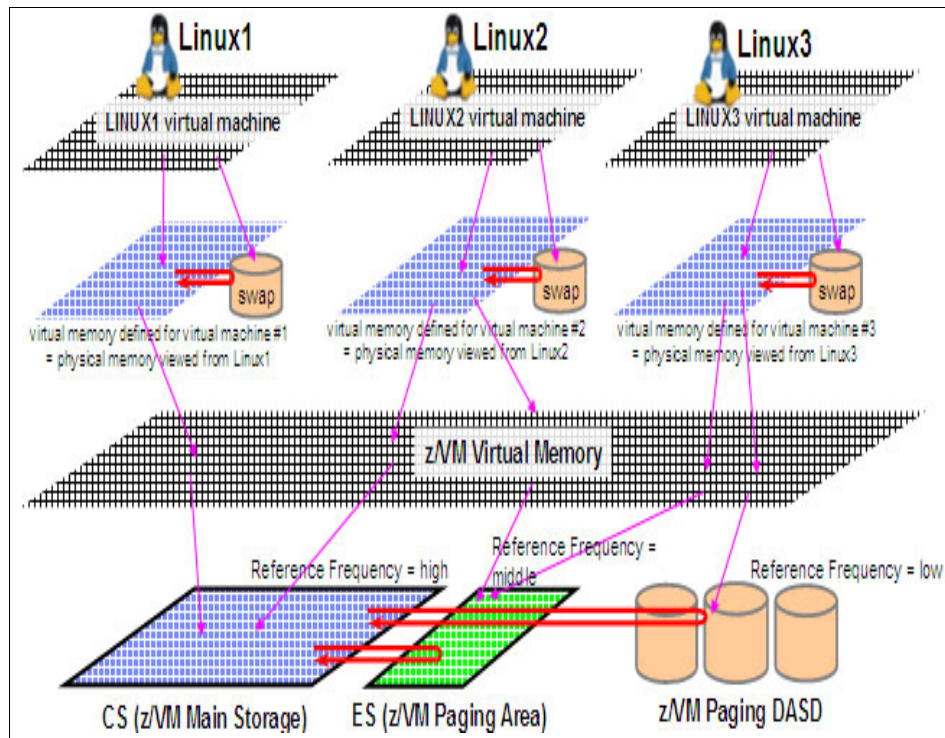


Figure 5-21 z/VM paging mechanism

In case of z/VM, there is a central storage area where the processes are executed. There is an expanded storage which is used for paging. DASD space is also used for paging. This is illustrated in Figure 5-21 on page 176. The CS in the figure indicate Central Storage and the ES indicate Expanded Storage.

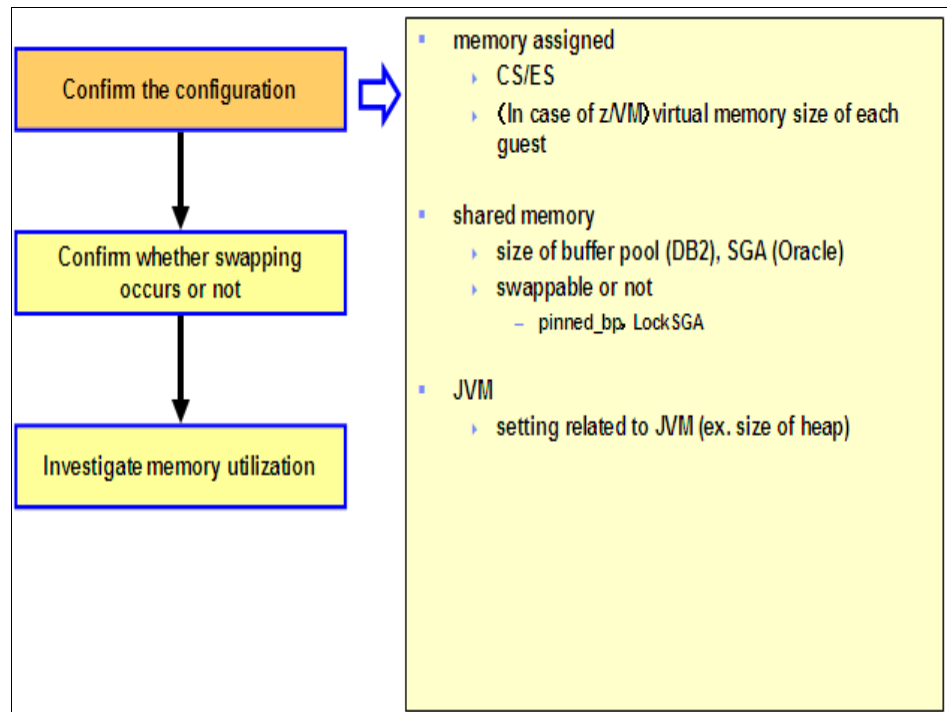


Figure 5-22 Memory bottleneck problem determination flow

Figure 5-22 on page 177 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 central storage, how much as expanded storage, virtual memory setting for each guest (z/VM), shared memory settings for middleware, etc are examined. When 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, etc. It is possible to control whether the shared memory should be swapped or not. The area in the memory where the data necessary for application execution is stored is called as the JVM™ heap in case of Java. Memory utilization of JVM cannot be viewed from the operating system and should be viewed from the JVM side.

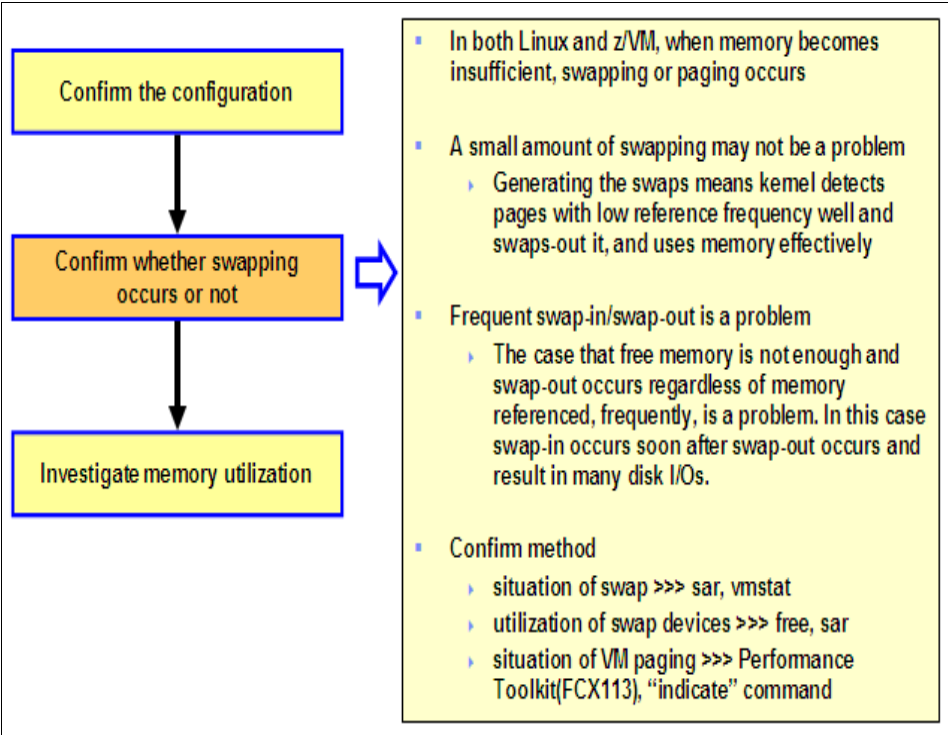


Figure 5-23 Problem determination flow - check for swapping

The next step is to check whether swapping occurs or not. Figure 5-23 on page 178 summarizes this. Figure 5-24 on page 178 illustrates how to check whether swapping occurs or not from the Linux side by using the command free.

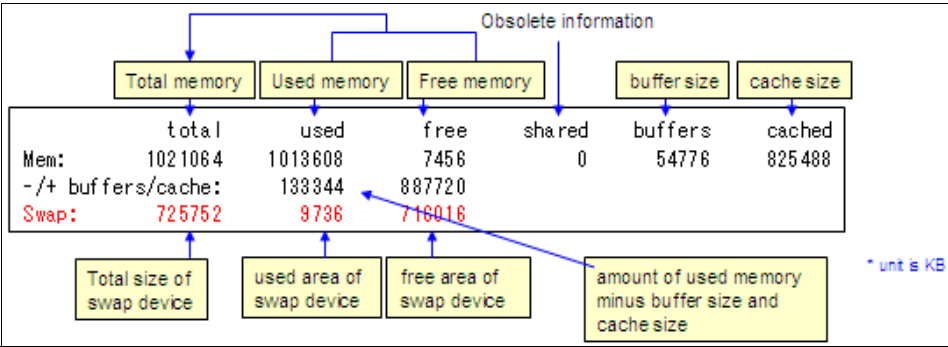


Figure 5-24 Confirming swapping by using free

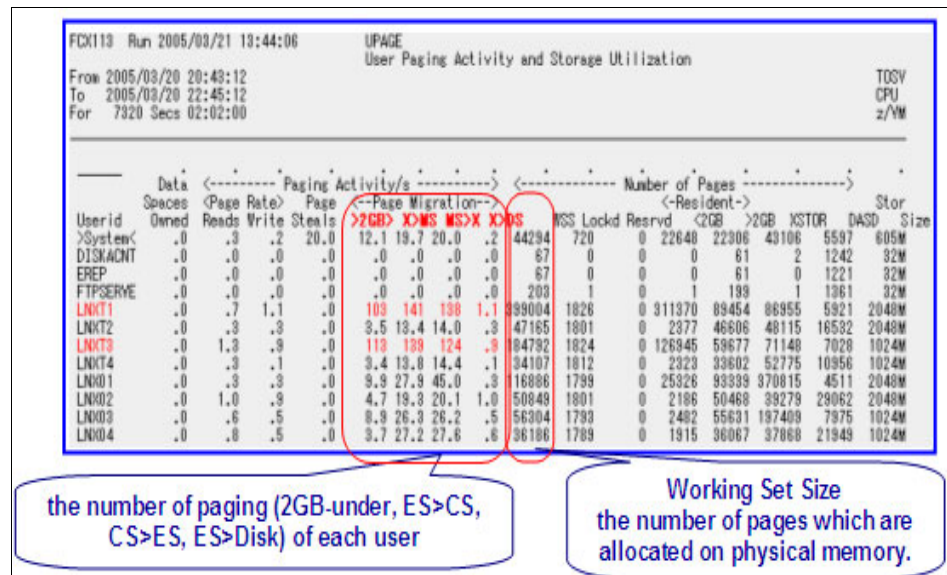


Figure 5-25 Confirming swapping by using z/VM Performance Toolkit

Figure 5-25 on page 179 illustrates how to check the paging activity from z/VM using the Performance Toolkit. Paging happens if there is a guest which gives overhead in the >2GB> memory area, when pages are migrated between the expanded and the main memory (X>MS and MS>X), when pages are migrated between the expanded memory area and the disk storage area (X>DS). Large values for the Working Set Size (WSS) will also cause paging.

The next step is of course, investigating where the memory is getting consumed. As shown in Figure 5-26 on page 180, the memory may be getting consumed by a user process, shared memory areas, JVM heaps, filesystem cache, etc. 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.

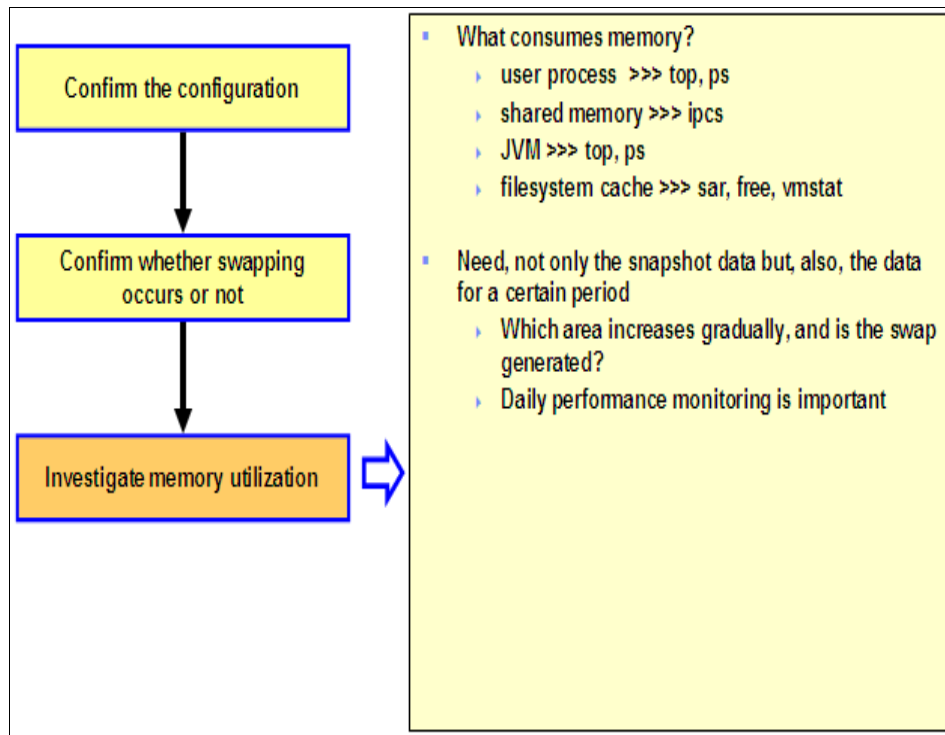


Figure 5-26 Problem determination flow - investigating the memory utilization

Figure 5-27 on page 181 shows a sample top output with the output data related to memory explained.

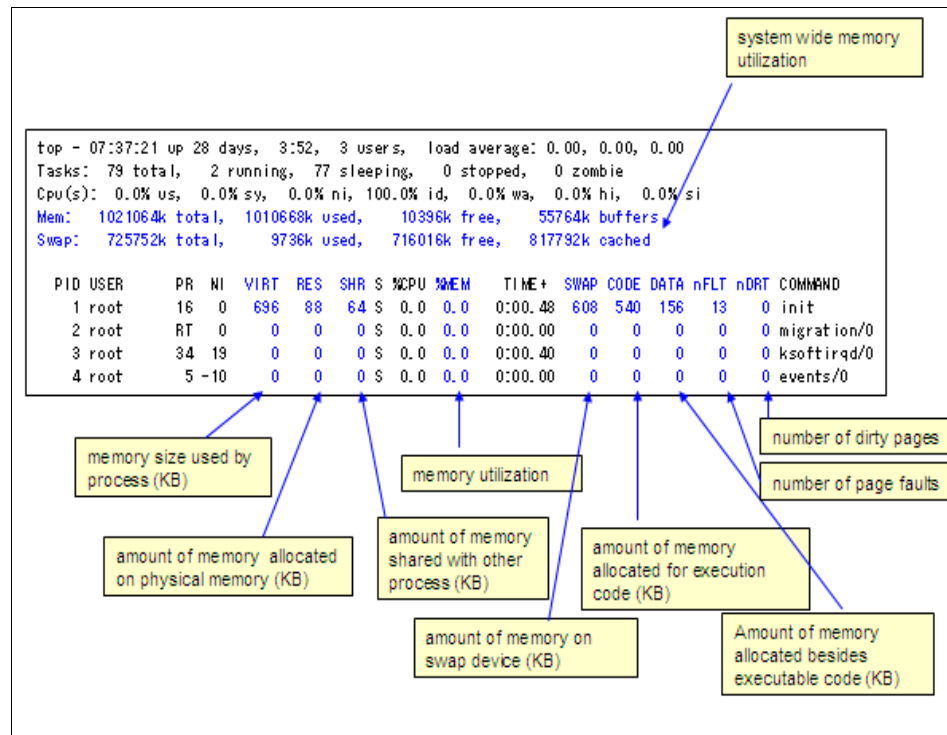


Figure 5-27 Checking memory utilization by using top

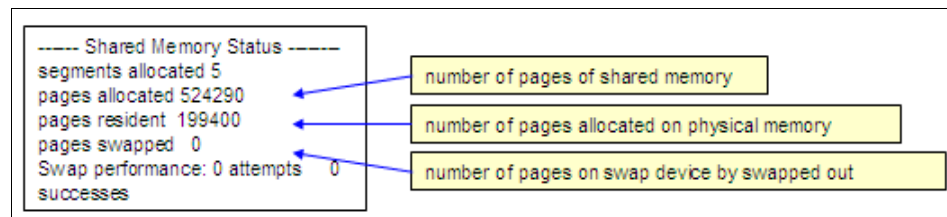


Figure 5-28 Checking memory utilization by using ipcs

Figure 5-28 on page 181 shows a sample ipcs output illustrating how to check the shared memory usage.

Linux uses the free memory as the filesystem cache. So, it is normal to find that Linux is not having any free memory if it is running for a long period of time. Such a cache gets released when a processes is executed.

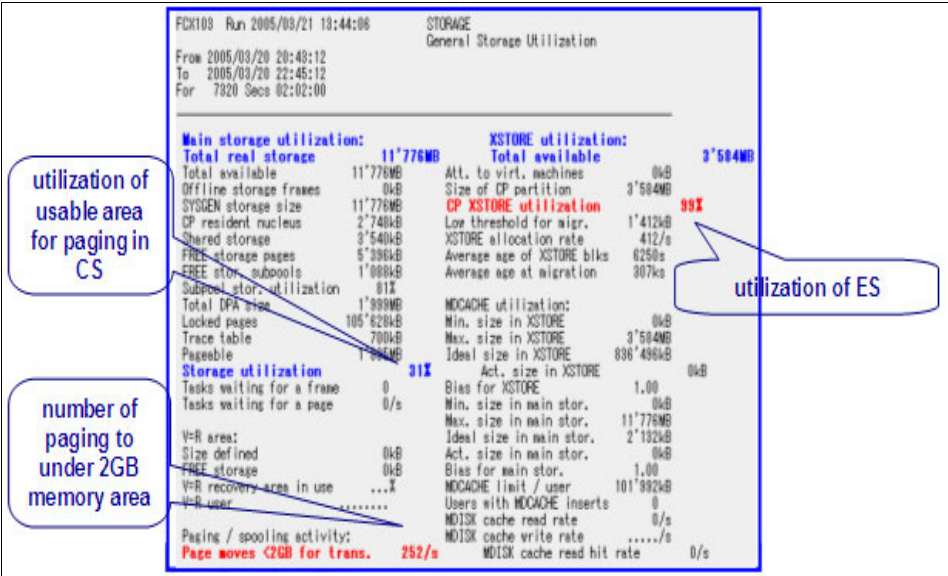


Figure 5-29 Confirming memory utilization by using z/VM Performance Toolkit

Figure 5-29 on page 182 shows a sample output z/VM Performance Toolkit indicating general stroage utilization. The output indicates the utilization of the usable area for paging in the Central Storage, the number of pages under the 2GB memory area and the utilization of the Expanded Storage besides many other detail. If the number of pages under the 2GB memory area is above 1000, then it should be investigated.

Sometimes it will be useful to acquire the monitoring data from the middleware. Figure 5-30 on page 183 shows sample monitoring information for DB2 and Oracle.

part of DB2 snapshot	
Buffer pool data logical reads	= 370
Buffer pool data physical reads	= 54
Buffer pool temporary data logical reads	= 0
Buffer pool temporary data physical reads	= 0
Buffer pool data writes	= 3
Buffer pool index logical reads	= 221
Buffer pool index physical reads	= 94
Buffer pool temporary index logical reads	= 0
Buffer pool temporary index physical reads	= 0
Total buffer pool read time (ms)	= 287
Total buffer pool write time (ms)	= 1
Asynchronous pool data page reads	= 9
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

part of Oracle Statspack				
SGA Target Size (M)	SGA Size Factor	Est DB Time (s)	Est DB Time Factor	Est Physical Reads
5,832	.3	237,747	1.0	5,245,722
11,264	.5	237,510	1.0	5,194,345
18,896	.8	237,488	1.0	5,189,874
22,628	1.0	237,488	1.0	5,189,874
28,180	1.3	237,488	1.0	5,189,874
33,792	1.5	237,488	1.0	5,189,874
38,424	1.8	237,387	1.0	5,183,207
45,056	2.0	237,387	1.0	5,183,207

SGA Memory Summary		DB/Inst: xxxxxxxx	Snap: 52-63
SGA regions	Begin Size (Bytes)	End Size (Bytes) (if different)	
Database Buffers	19,280,243,808		
Fixed Size	2,113,896		
Redo Buffers	14,859,584		
Variable Size	4,345,302,880		
sum	23,822,320,128		

Figure 5-30 Checking memory utilization from middleware

5.2.4 I/O bottlenecks

The I/O bottlenecks are either related to the disk subsystem or to the network subsystem. We examine each of them in the following sections..

Disk bottlenecks

.When the CPU is waiting long for the disk I/O operations to finish, we experience the disk I/O bottleneck. This may happen if there are too many disk I/O operations happening or the disk operations became too slow. Needless to say, the disk bottleneck affects the throughput and response time of the system. As shown in Figure 5-31 on page 184, as usual, the problem determination starts by checking the current configuration. It is followed by confirming whether I/O wait is occurring in the system or not. The storage subsystem configuration, the System z configuration and the Linux level configuration are the things to examine. The storage subsystem configuration details include the subsystem type/model, number of ranks, number of DA pairs, size of the cache, configuration of the cluster, etc..

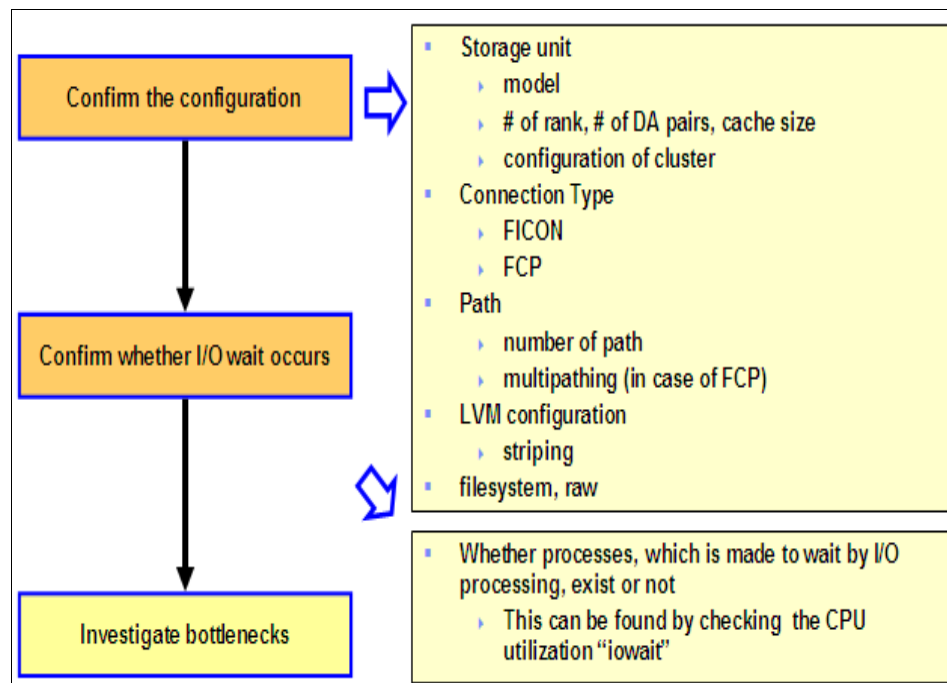


Figure 5-31 Disk bottleneck problem determination flow

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 (i.e. FICON or FCP) needs to be determined. Number of paths for the connection from the System z server to the storage subsystem is also important. In case of FCP connection, multipathing information to be gathered. At the Linux side, the Logical Volume Manager (LVM) configuration to be examined, if in use. Also, information about the type of filesystems or partition used needs to be gathered.

Figure 5-32 on page 185 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.

vmstat															
procs		memory				swap		io		system			cpu		
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa
1	7	4	3028	7820	414784	0	0	4	110192	151	344	0	45	0	55
0	3	4	4764	7848	411144	0	0	0	35696	143	140	0	15	0	85
0	3	4	4532	7884	412656	0	0	0	18560	147	157	0	15	0	85
0	3	4	3912	7896	414192	0	0	4	4108	145	121	0	8	0	92
2	2	4	3060	7920	416748	0	0	0	22720	146	5869	1	12	0	87

CPU utilization
"iowait"

number of process waiting for
I/O processing

Figure 5-32 Examining disk bottleneck by using vmstat

Figure 5-33 on page 185 indicates the points to be considered for investigation at the Linux side, the System z server side and the storage subsystem side.

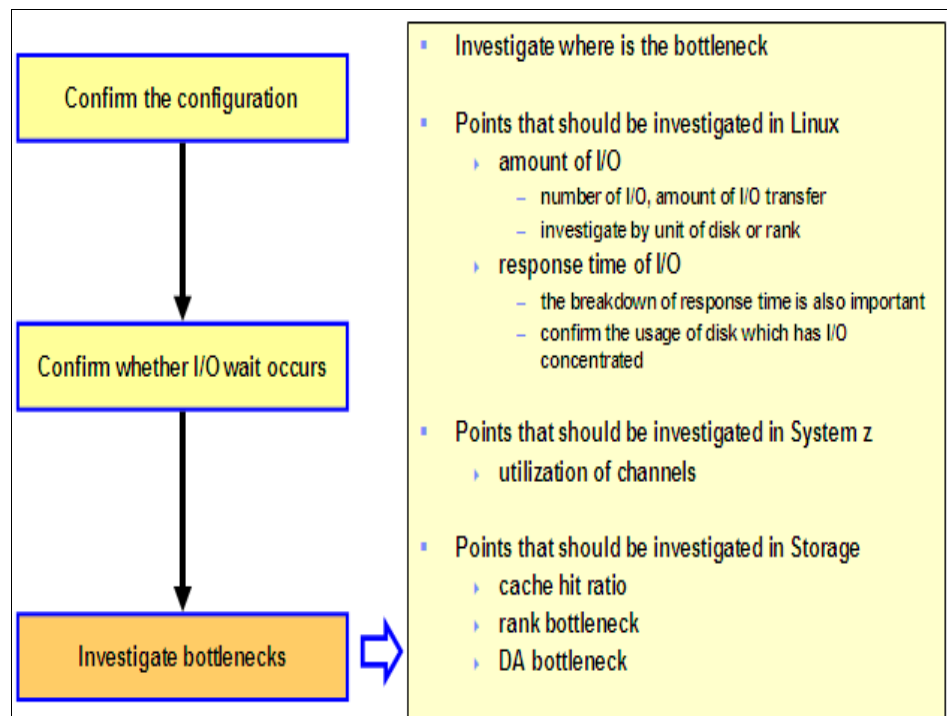


Figure 5-33 Problem determination - investigate the disk bottlenecks

Figure 5-34 on page 186 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”. Amount of data transfer corresponds to “rkB/s” and “wkB/s”. Average response time corresponds to “await” and “svctm”. If there is a big difference between “await” and “svctm”, then it means that there is lot of time spent in the device driver queue and channel is busy. If the “svctm” value is high, then it means that 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.

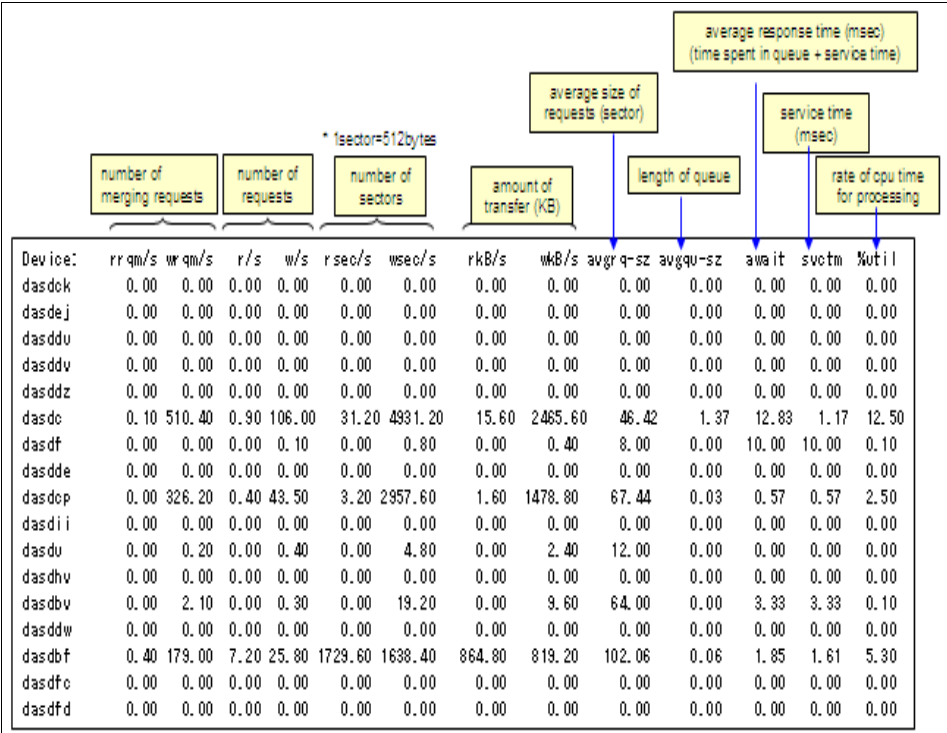


Figure 5-34 Investigating the disk bottlenecks by using iostat

Figure 5-35 on page 187 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 time in the “Time (msec)” column is expected to be nearly identical under normal circumstances. If the value of “%Busy” exceeds 50, then that means there is a performance issue and the response time is high. For good performance, this value should be below 30.

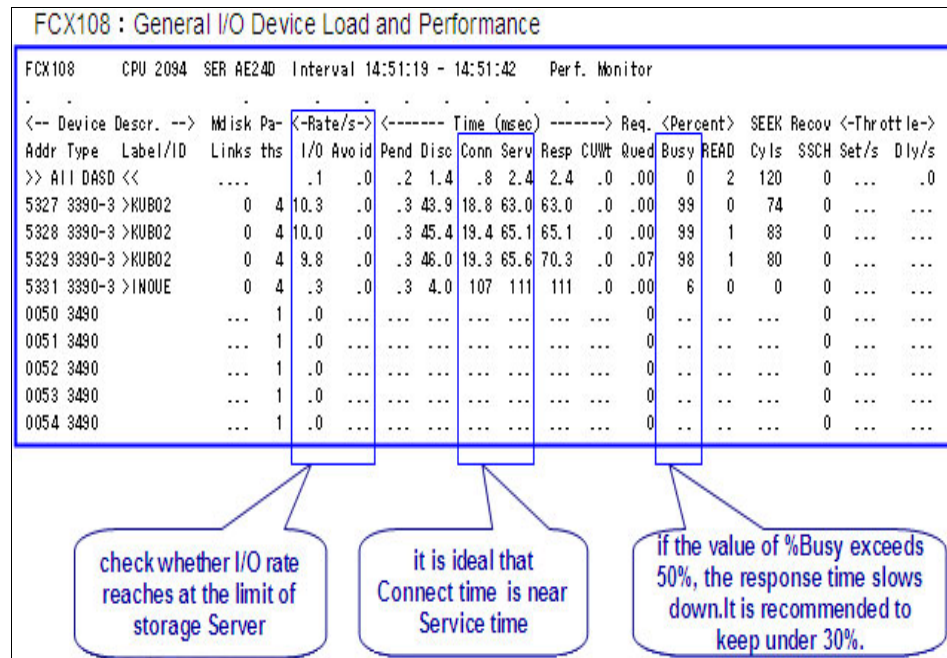


Figure 5-35 I/O load and performance check by using the z/VM Performance Toolkit

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

and can be deactivated with:

echo set off > /proc/dasd/statistics

Figure 5-36 on page 188 shows a sample DASD statistics. DASD statistics is stored in the /proc/dasd/statistics file.

01	50881	dead	I/O	requests															
02	with	5270818	sectors	(512B	each)														
03	_4	_8	_16	_32	_64	_128	_256	_512	_1k	_2k	_4k	_8k	_16k	_32k	_64k	128k			
04	_256	_512	_1N	_2N	_4N	_8N	_16N	_32N	_64N	128N	256N	512N	_16	_26	_46	_>46			
05	Histogram	of	size	(512B	secs)														
06	0	0	1039	4799	8102	38957	4475	292	195	1422	0	0	0	0	0	0			
07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
08	Histogram	of	I/O	times	(microseconds)														
09	0	0	0	0	0	0	0	0	2	8	109	3244	29570	17480	7888	1248			
10	1390	193	11	0	0	0	0	0	0	0	0	0	0	0	0	0			
11	Histogram	of	I/O	times	per	sector													
12	0	0	0	0	0	178	4141	24084	15839	9508	2513	801	173	41	7	0			
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
14	Histogram	of	I/O	time	till	ssch													
15	5	1	2	0	0	0	0	0	2	4	301	11527	29339	12278	5156	1759			
16	383	118	8	0	0	0	0	0	0	0	0	0	0	0	0	0			
17	Histogram	of	I/O	time	between	ssch	and	irq											
18	0	0	0	0	0	0	0	0	2584	23898	18720	5307	2325	2725	1217	82			
19	23	21	1	0	0	0	0	0	0	0	0	0	0	0	0	0			

Figure 5-36 Sample DASD statistics

Figure 5-37 on page 188 shows the histogram of I/O times for the DASD statistics.

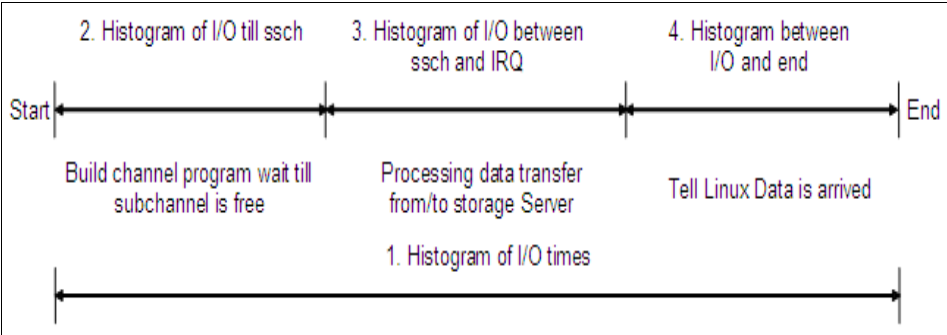


Figure 5-37 DASD statistics - Histogram for I/O times

Refer to “DASD Statistics” on page 67 for a detailed explanation. Figure 5-38 on page 189 shows an example analysis of the DASD statistics data.

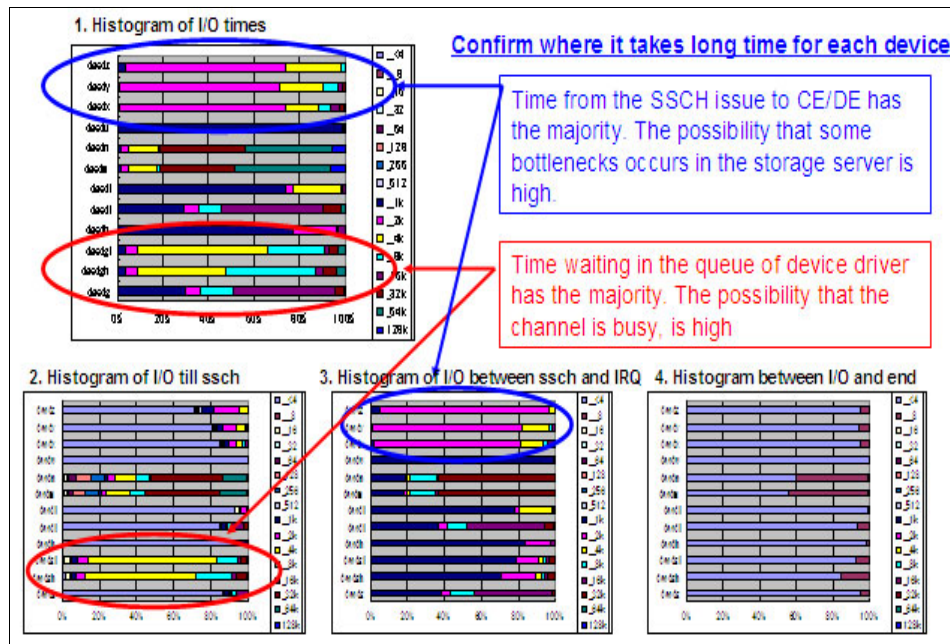


Figure 5-38 DASD statistics analysis example

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 on page 190 shows the breakdown of the SCSI latency that can be acquired using the zfcps statistics. Refer to “zFCP/SCSI” on page 103 for a more detailed explanation. Table 5-3 on page 190 indicates the conclusions to derive for an HBA based on the zfcps statistics.

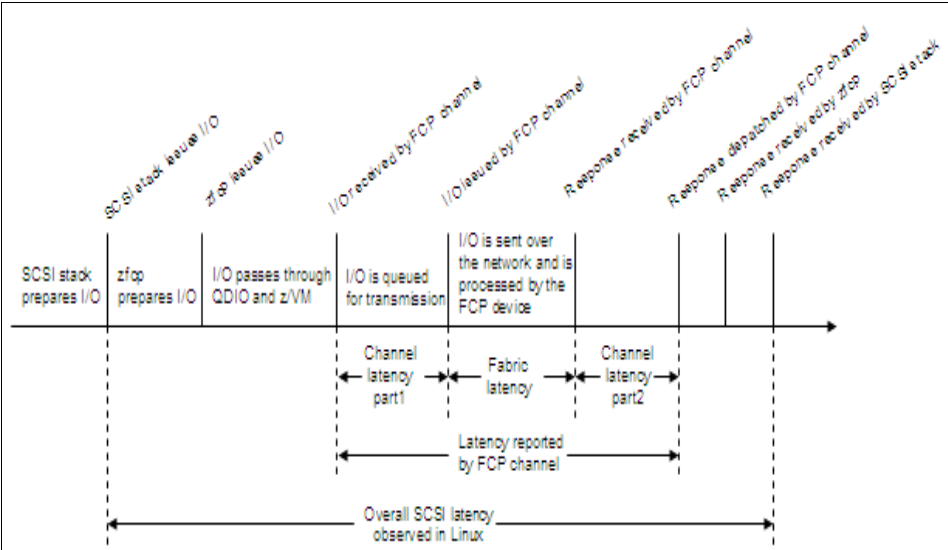


Figure 5-39 Breakdown of SCSI latency

Table 5-3 on page 190 lists the some of the things to look for in the host bus adapter based on the zfcpl 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 on page 191 lists the some of the things to look for in the logical unit based on the zfcpl 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 zfcpl 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 on page 192 shows a sample output of the z/VM Performance Toolkit.

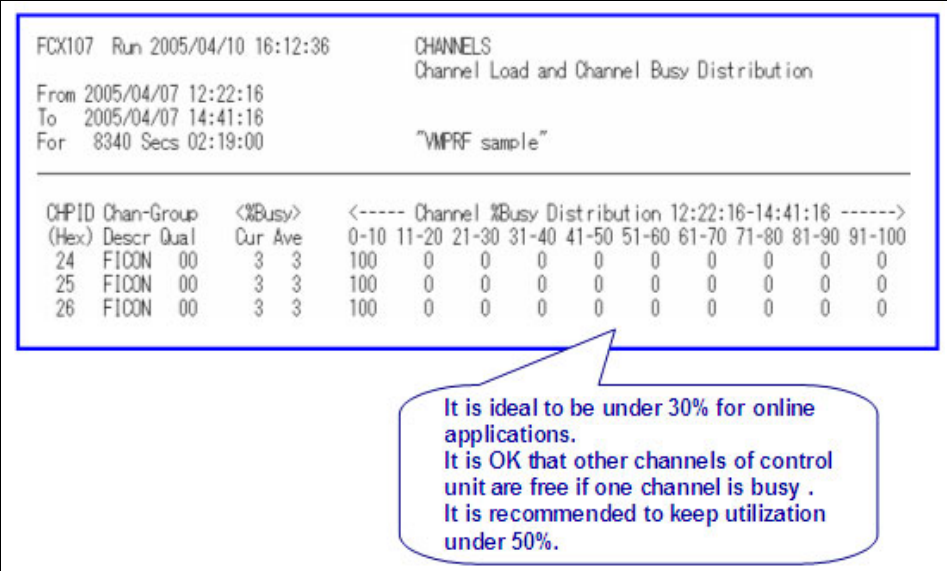


Figure 5-40 Investigating channel bottleneck by using the z/VM Performance Toolkit

This output indicates the channel load and channel busy distribution. As noted, it is suggested to keep 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, the DA pair level, HBA level, etc. The cache and processor used in the storage subsystem may also show a bottleneck at times depending on the load it is carrying. In case of sequential I/O, the interest should be in measuring the throughput and in 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, etc. 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 on page 193 and Figure 5-42 on page 194 show the different steps involved in the problem determination of network I/O performance. Type of the networking option used (such as OSA Express, HiperSockets, VSWITCH/Guest LAN), the way these networking options are configured (eg: Layer 2 or Layer 3, Channel Bonding, MTU size, etc) and the network environment to which the system is connected are the main things to be noted while checking the current configuration.

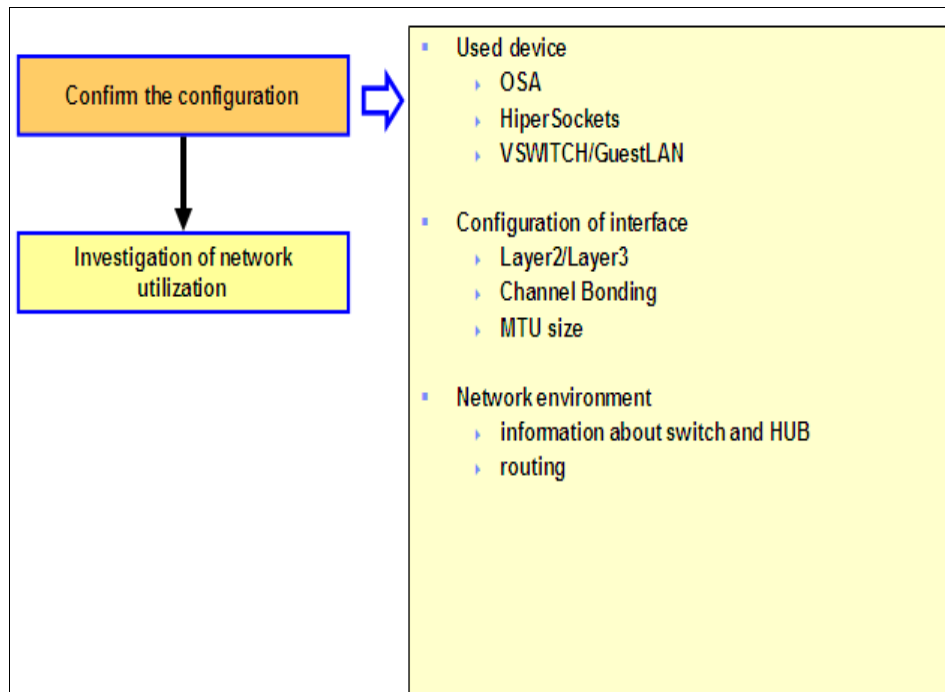


Figure 5-41 Network performance problem determination flow

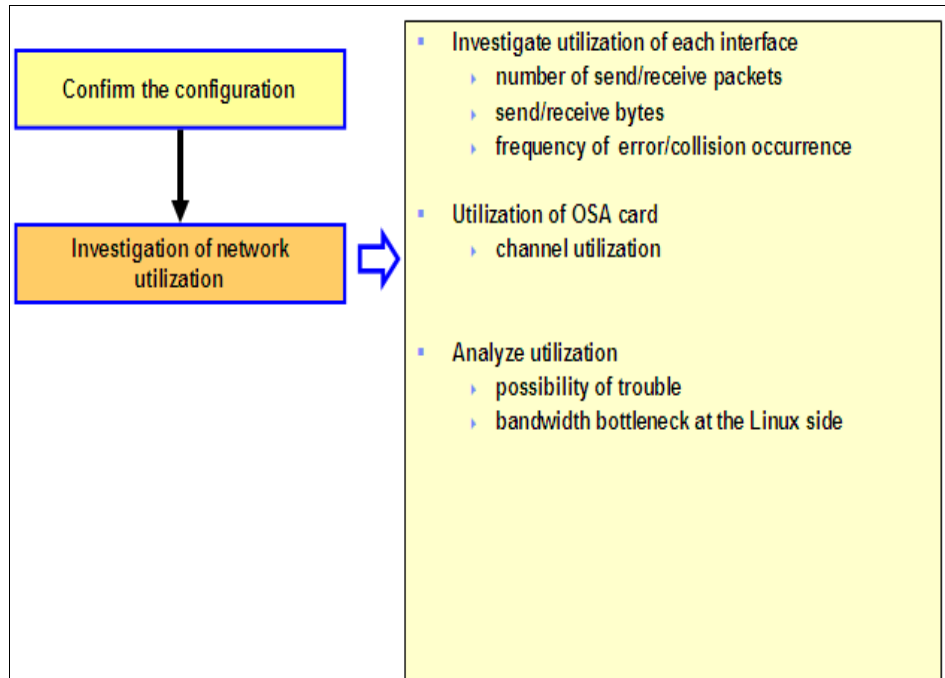


Figure 5-42 Network performance problem determination - investigating utilization

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, etc. 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 case of an OSA card, the channel utilization to be examined. Investigate whether the bandwidth of OSA or HiperSockets becomes full by referring performance reports. Things to check at the Linux side are the bandwidth bottleneck, incorrect configurations, etc.

Note: 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 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, etc.

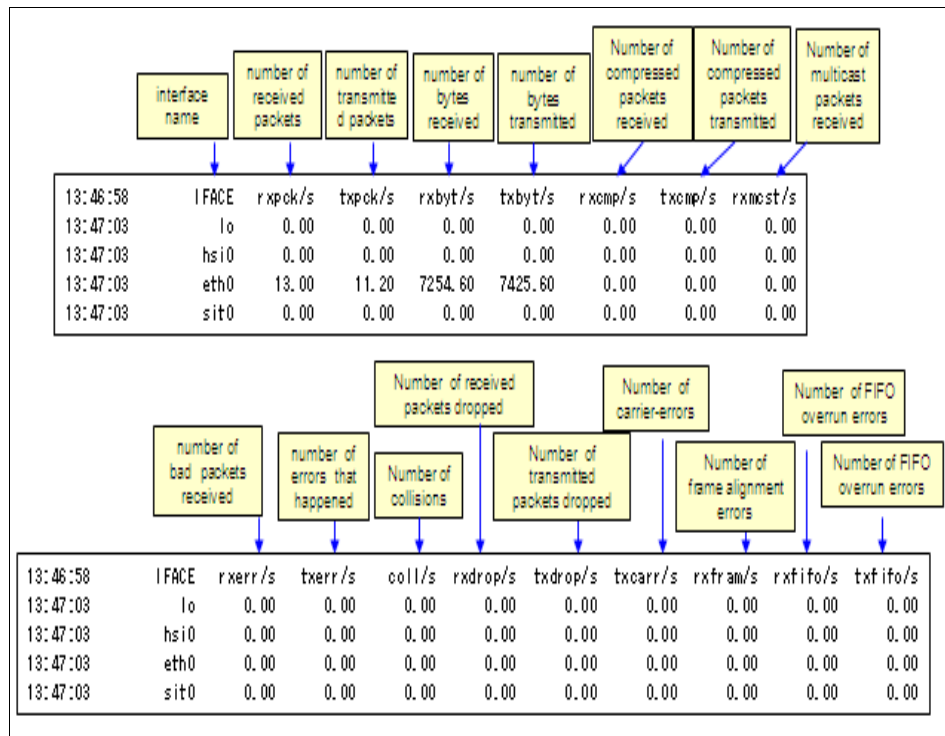


Figure 5-43 Network utilization check by using sar

Figure 5-43 on page 195 shows a sample sar output useful for investigating network utilization. Figure 5-44 on page 196 shows three screen from the z/VM Performance Toolkit viz; the HiperSockets channel activity screen (FCX231), the Virtual Switch channel activity screen (FCX240) and the QDIO device activity screen (FCX251).

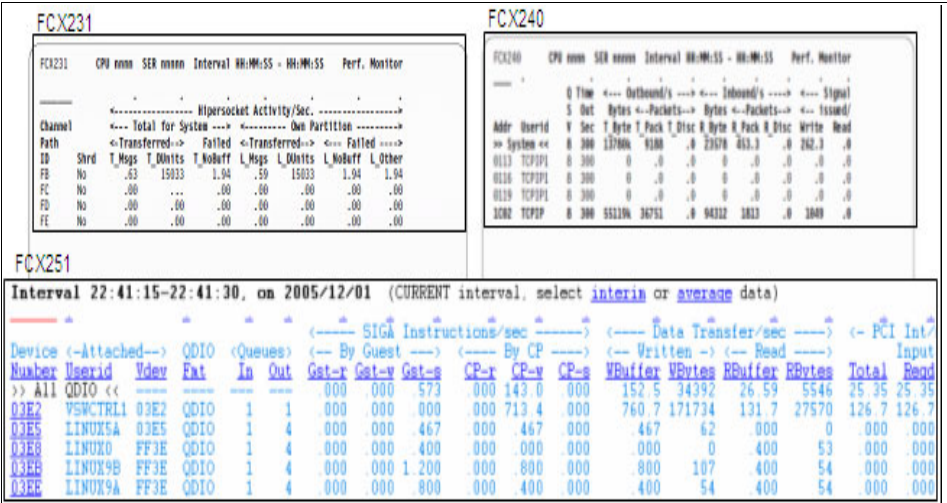


Figure 5-44 Network utilization by using the z/VM Performance Toolkit

Sometimes it is necessary to examine the data acquired from the Linux side in conjunction with the z/VM Performance Toolkit data to identify the bottleneck.

5.2.5 Case studies

Case study 1 - slow batch processing

In this case study we will examine a problem reported as slow batch processing.

Problem

One of the batch processing job is not finishing by the time by which it is scheduled to finish.

Data for analysis

To troubleshoot the problem, we need to gather the following information and analyze.

- ▶ sar output
- ▶ vmstat output
- ▶ iostat output

Analysis and conclusion

Figure 5-45 on page 197 shows the graph preapred 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.

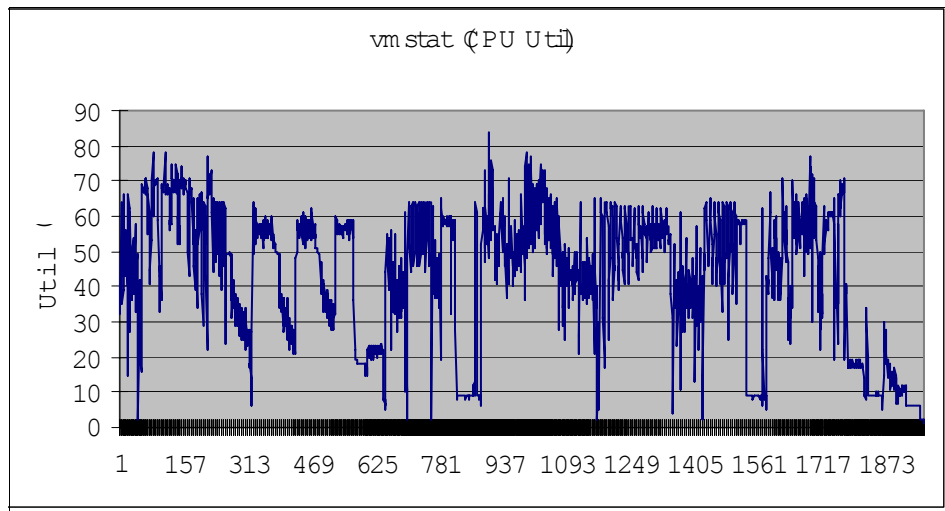


Figure 5-45 CPU utilization report based on the vmstat output

Figure 5-46 on page 197 shows the graph plotted using the “Run Queue” data from the vmstat output. This system is having 16 CPUs, so at any point in time,

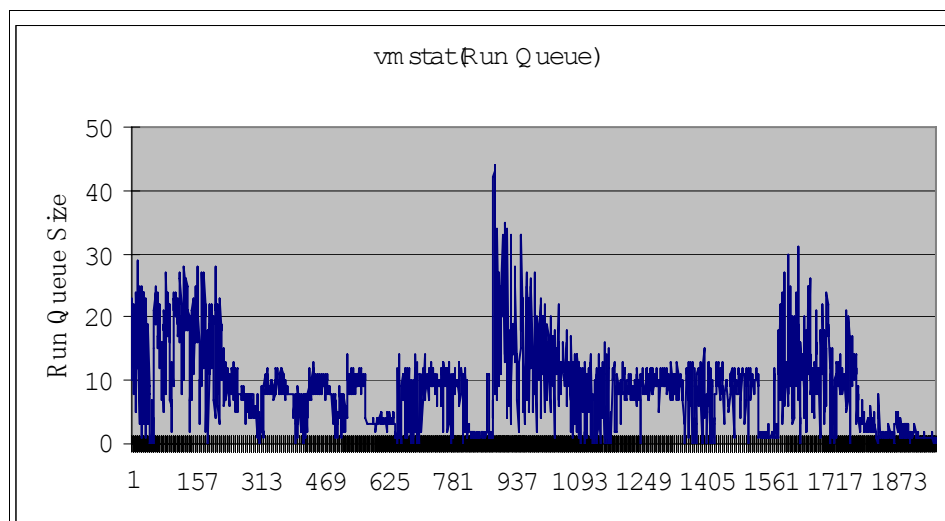


Figure 5-46 Graph based on the vmstat Run Queue data

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 5-47 on page 198 shows an example of improved CPU utilization.

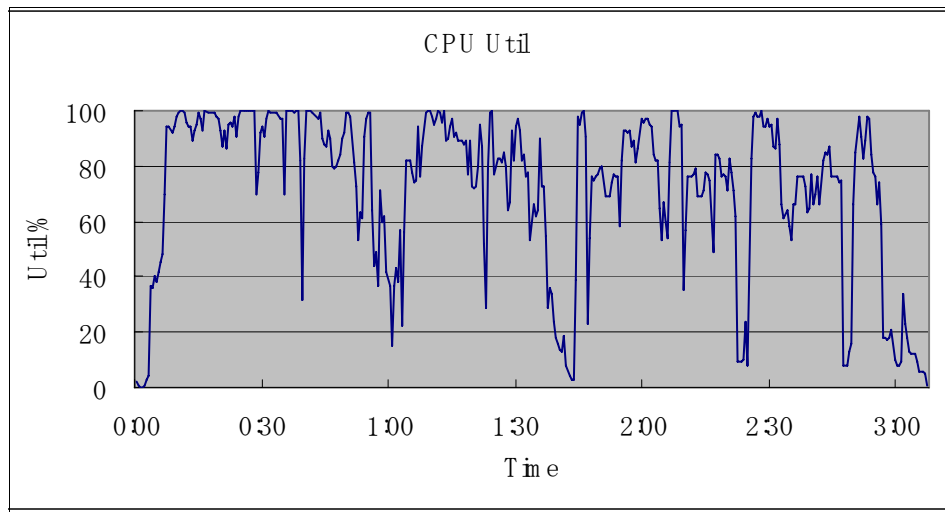


Figure 5-47 Improved CPU utilization

This is a classic case where we experience that a performance bottleneck can arise even when all the resources on the system are not used too.

Case study 2 - Slow database server response

In this case study we will examine a problem reported as 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 the following information and analyze.

- ▶ 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 on page 199 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 log buffer into online redo log files when dirty blocks in the dirty buffer are written on to 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, that logical volume is not striped. Because of this reason, 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 will 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 the following information and analyze.

- ▶ 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%, iowait: 5%. Since the lion’s share 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 on page 200. So, the function “aaaaaaaa” in the library “AAAAAA.so.1” provided by the middleware is called very 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	aaaaaaaa
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	cccccccc
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

6



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, the following topics are discussed:

- ▶ How to determine a storage related problem
- ▶ Case study

6.1 How to determine a storage related problem

To determine if the problem you encounter on your Linux system is related to storage, you need to know some commands to investigate the storage devices. In this section we discuss the different commands to use for the different types of storage devices.

To get an overview about your storage devices use the `lscss` command. It 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`, lists all channel path (CHP) information whether it's online or not. The status of the channel subsystem can also be checked.

In Figure 6-1 on page 202 you can see the `lscss` command with some explanation.

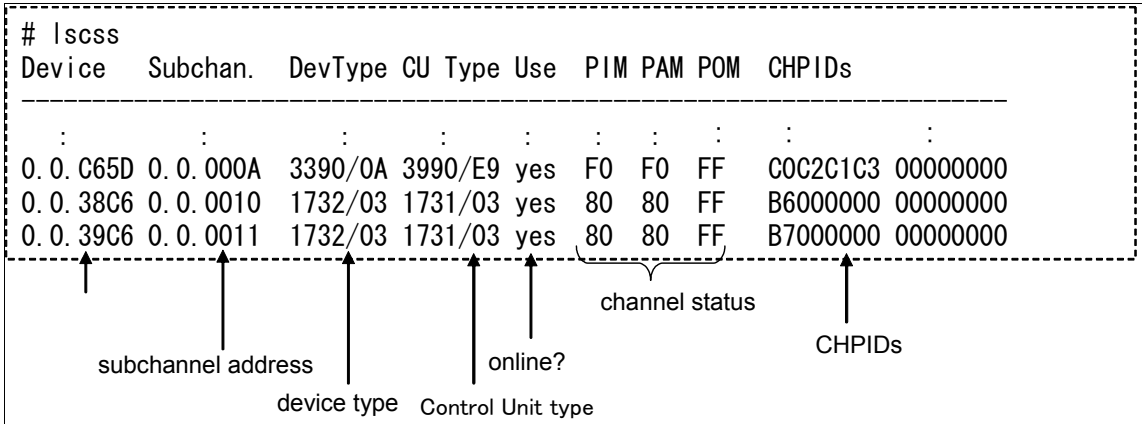


Figure 6-1 `lscss` command

If a device is not listed with the `lscss` command you should check the following:

- ▶ Is the IOCP definition correct?
- ▶ Is the target device correct defined in z/VM?
- ▶ Check the parameter "cio_ignore" (More details in ...)

If the address of the target device is listed with the `lscss` command but the "Use" column is blank you should check:

Depending on the way the devices are brought online,

- ▶ via kernel parameter in `zipl.conf`
 - check the `zipl.conf` file
 - after editing the `zipl.conf`, run the `zipl` command

- ▶ via initrd
 - check initrd, then run **zipl**
 - 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 etc.

What is 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.

Figure 6-2 on page 203 shows how to add and delete devices using the **cio_ignore** command.

Example 1 (set up in zipl.conf)

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

(devices 1000 – 1FFFF, 3000 – 3FFF are not detected)

Example 2 (set up with /proc/cio)ignore dynamically)

```
# echo add 0.0.1000 > /proc/cio_ignore
```

(device 1000 is added to cio_ignore list)

```
# echo free 0.0.2000 > /proc/cio_ignore
```

(device 2000 is removed from cio_ignore list)

Figure 6-2 How to add and delete devices using the cio_ignore parameter

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 6-3 on page 204 you see the **lsdasd** command with the response from the system and some explanation on how to interpret the system messages.

```
# lsdasd
0.0.5481 (ECKD) at ( 94: 0) is dasda      : active at blocksize 4096, 1803060 blocks, 7043 MB
0.0.c65f (ECKD) at ( 94: 4) is dasdb      : n/f
```

device address major/minor number device name device status

- "active" means available
- "n/f" means unformatted -> you should dasdfmt

Figure 6-3 *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 use format 1 for the dump (default)
 - 2 use format 2 for the dump
- 'begin' and 'size' can have the following format:
x[klmlblt|c]
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 content (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 6-4 on page 205 and in Figure 6-5 on page 205.

```
# 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 6-4 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 6-5 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 6-6 on page 206.

```
# 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 6-6 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 this devices. This command reads the status from sysfs and displays all SCSI devices that are online.

Figure 6-7 on page 206 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]
```

SCSI address device type maker product code firmware version device name path in sysfs

Figure 6-7 lsscsi command

If the target address is not displayed with **lsscsi**, you may want to check the following things, depending on the way the disk was brought online:

- ▶ with initrd
 - after modifying initrd, run zipl
 - check linuxrc script in initrd (to verify the procedures)
- ▶ coldplug
 - check /etc/sysconfig/hardware/hwcfg-*
 - especially, WWPN and LUN ID

Check if the target LUNs are mapped? Use `san_disc` to verify:

- ▶ SAN switch zoning
- ▶ LUN mapping
- ▶ cabling

With the command `lszfcp` you can get more information about zfc adapters, ports, and units that are online. This function becomes available with SLES10 only. (check this!!) In Figure 6-8 on page 207 you can see an example of a `lszfcp` command.

```
# lszfcp -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 6-8 `lszfcp` 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. Please see an example of the `san_disc` command in Figure 6-9 on page 208.

```
# 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
```

ports on storage connected to the specific port on System z

lunids mapped to the specific port on storage

Figure 6-9 *san_disc* command

6.1.3 What to check for multipathing

The command `multipath -ll` reads the sysfs and displays the multipath devices.

```
# 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]
```

Alias

SCSI devices

wwid

Figure 6-10 *multipath -ll* command

`/etc/init.d/multipathd` status checks the status of `multipathd`:

```
# /etc/init.d/multipathd status
Checking for multipathd:      running
```

Figure 6-11 */etc/init.d/multipathd*

The path checker's log is stored to /var/log/messages

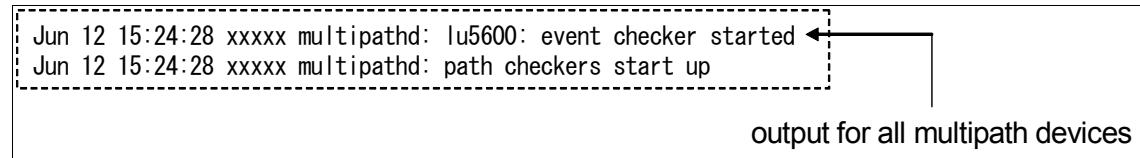


Figure 6-12 path checker

What to check if in case that the target disk is displayed with multipath -ll:

Disk must be varied online in initrd for multipathing on booting

- ▶ If multipathing can be configured with multipath command after booting, disks may be brought online in coldplug
 - Check linuxrc script in initrd
 - 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 device name

When configuring multipathing, the files named alias is 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 6-13 on page 210.

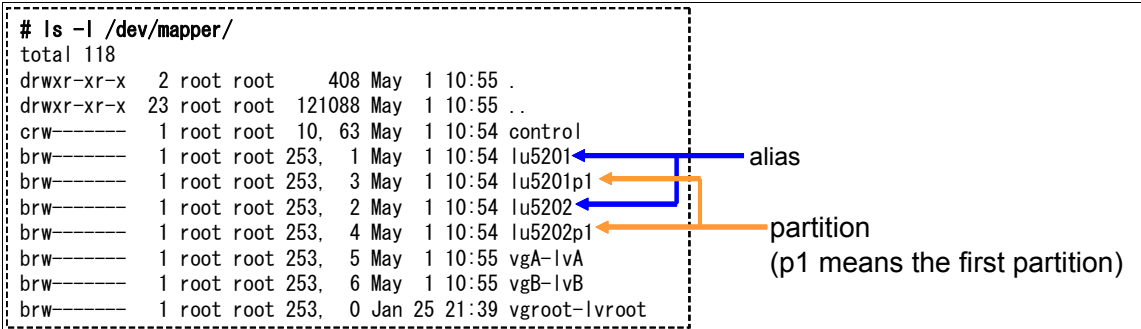


Figure 6-13 /dev/mapper

6.1.4 What to check for LVM

When you are using LVM in your Linux sysgtem, here are some things you should check incase of problems within the LVM:

How to check LVM recognition:

With the command `vgdisplay -v` you get some useful information about the LVM volumegroup. An example of the `vgdisplay -v` command is shown in Figure 6-14 on page 210 and continued in Figure 6-15 on page 211.

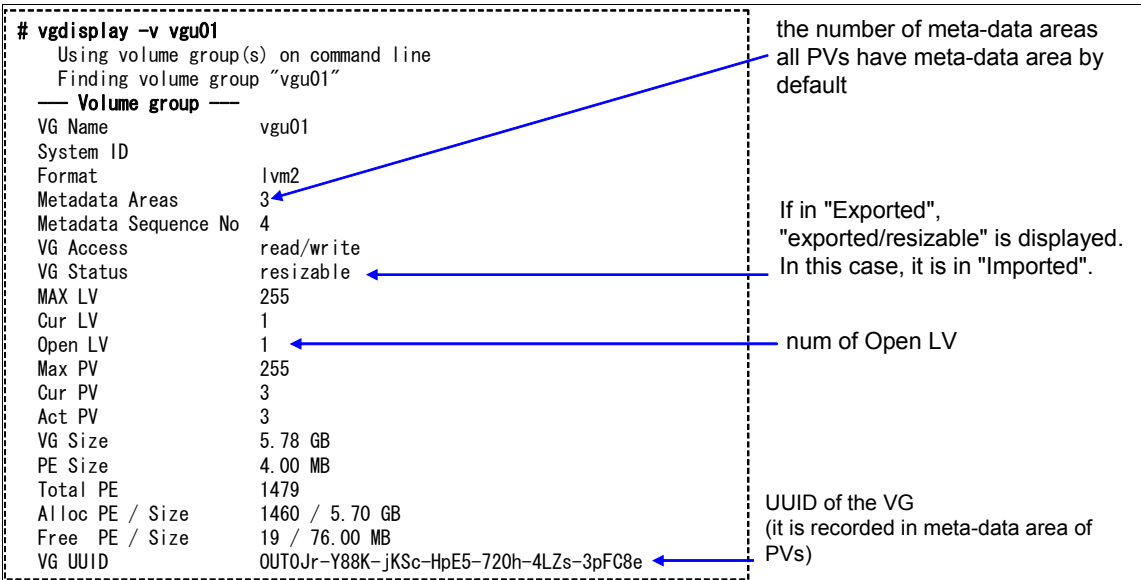
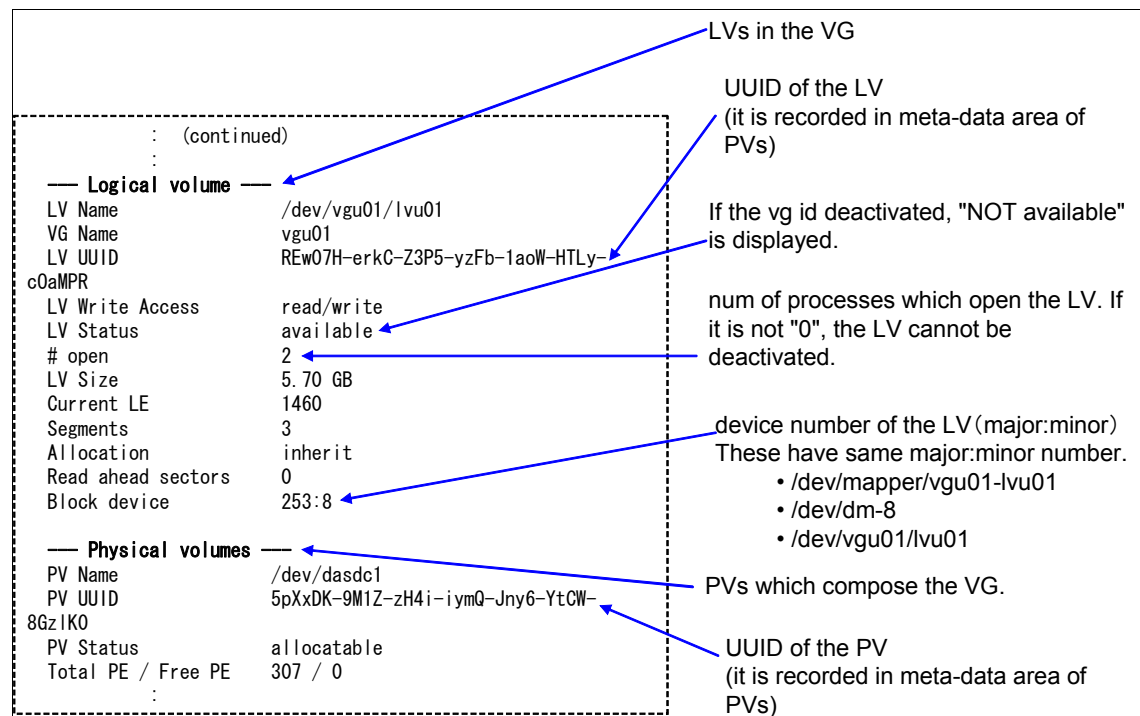


Figure 6-14 vgdisplay -v

Figure 6-15 `vgdisplay -v` (continued)

With the `vgscan` command, all scanned devices are stored in the `./etc/lvm/.cache` file. 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 6-16 *lvm.conf* file

The recognition of the LVM by the vgscan command is the following sequence:

PVs -> VGs -> LVs

Unless all PVs which are part of a VG are recognised, the VG is not available. First PVs need to be recognized in vgscan while booting. PVs need to be recognized in initrd or ziplt.conf.

The pvdisplay command is available to display the recognized PVs

In case that PVs are recognized but the VG is not recognized the meta-data area in a PV may be broken.

In 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 into a file for further analysis. You can switch the DASD I/O statistics on and off as needed. Please see Figure 6-17 on page 213 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 6-17 Switch DASD I/O statistic on/off

To reset the collection switch DASD I/O statistics off and on again.

Please see also Chapter 3.1.8, “netstat” on page 66 for more information 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 will discuss a scenario where the swap rate will become too high and the system throughput will decrease dramatically.

We have a Linux guest system which runs a DB2 database. The Linux system has an increase its workload due to some DB2 requests. We used z/VM tools and Linux basic tools to monitor this situation. In the first screenshot (see Figure 6-18 on page 215) of the performance toolkit screen 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 6-18 IBM z/vM Performance Toolkit shows normal load and paging activity

The next screen (Figure 6-19 on page 216) shows the detailed view of the Linux guest system running the DB2 workload, LNXDB2. You can see that in this stage, there is a relatively high I/O rate but no problem is detected.

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:/s	XSTORE dedic.:	0MB	XSTORE > DASD : .0/s
I/O rate	: 222/s	XSTORE pages	: 70160	SPOOL pg reads : .0/s
DASD IO rate:	.0/s	DASD slots	: 52619	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%
Stacked blk	: ..	Page wait	: 0%	CF wait : 0%
Stat.: EME,P02,PSWT		I/O wait	: 0%	Sim. wait: 1%
				I/O act. : 0%
				Eligible : 0%
				Runnable : 1%
Proc.	%CPU	%CP	%EM	%VECT
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

Figure 6-19 Detailed view of the Linux guest system

In the next screenshot of the overall system performance screen 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				LNWDB2	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													



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				



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				

--

--

--

--

Now the XSTOR paging increased dramatically to 1556 per second. This means that the overall system throughput is very low since the system 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 %		LNxWAS	16.9											
Trans. per sec.				1.4	1.9	.5	Max. VECT %												
Av. time (sec)				.088	.014	.222	Max. IO/sec		LNxWAS	1.6											
UP trans. time					.014	.222	Max. PGS/s		LNxSU3	29.7											
MP trans. time					.000	.000	Max. RESPG		LNxWAS	763955											
System ITR (trans. per sec. tot. CPU)						12.9	Max. MDCIO												
Emul. ITR (trans. per sec. emul. CPU)						.0	Max. XSTORE		LNxDB2	95945											
Command ==>																					
F1=Help F4=Top F5=Bot F7=Bkwd F8=Fwd F12=Return																					

As result of this high workload, shortage of memory and extremly high paging activity the system is in a so called emergency scanning mode. This can be seen in the next screenshot. Please see the field ESCN 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														

Note: See also chapter 11.4 for more information about emergency scanning.

In the last screen you see a summary of the memory usage of userids 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.



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 >Resrvd	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										

→



e: You can also use the **Toolkit** → **Quick Access Menu** to issue the **Toolkit** → **RXFM** → **Insert_tool** commands



7

Eligibility Lists

This chapter discusses about CP scheduling process and z/VM tools for problem determination. Also we discuss the corective measure we need to take to over come the problem with respect to the eligibility lists.

7.1 CP Scheduler

When you log on to z/VM, CP creates a control block1 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.

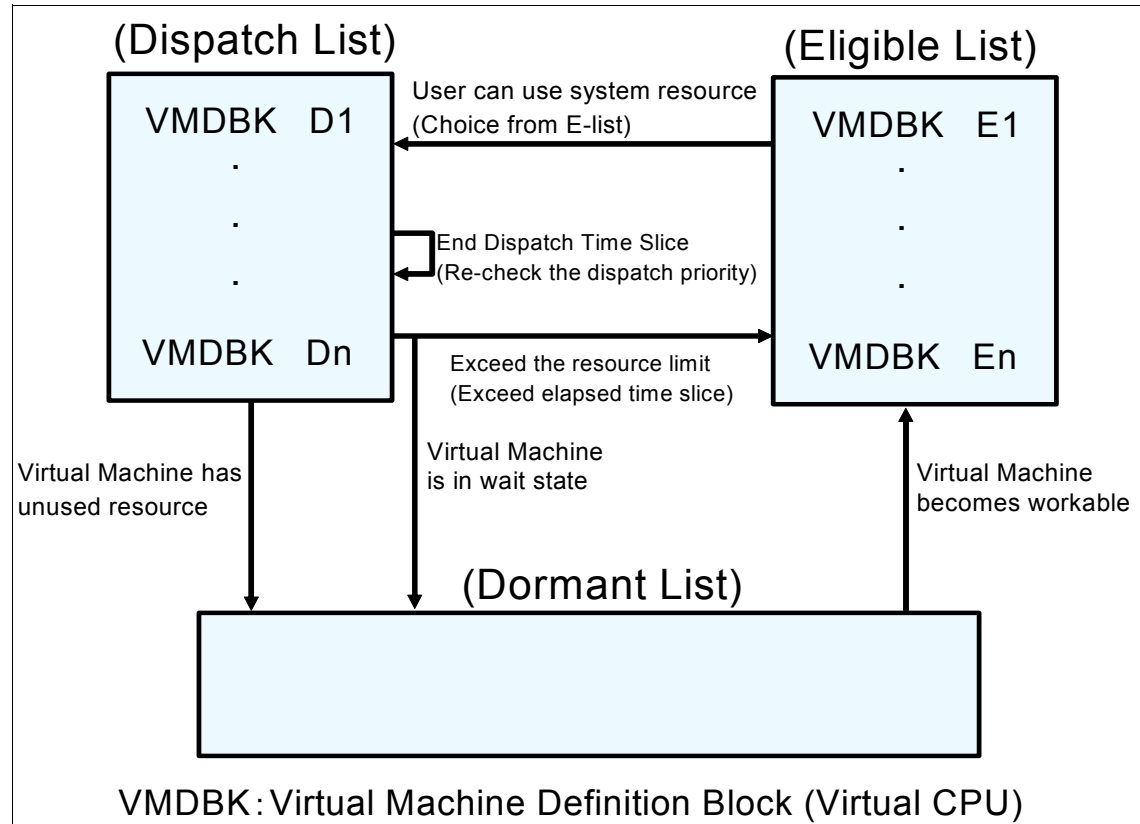


Figure 7-1 z/VM Scheduler lists

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 that are idle (as far as we know).

If your virtual machine is not doing anything, then CP will place it in the dormant list, which implies that your virtual machine is “asleep”. If you “wake 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 some system resource. Before it is given any resource, 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 timeslice. If you have more work to do, you can stay in the dispatch list and get another time slice until either 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 really do need some more processing capacity. You will then be upgraded to be a class 2 user and allowed to stay in the dispatch list for longer intervals to see if 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 if 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 that have just begun a transaction, and therefore are assumed to be currently processing short transactions.
Class 2	This class indicates the users that 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 that 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 Figure below. 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
```

7.1.1 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 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 way to solve the Thrashing problem is to set SRM STROBUF commad. Basically the SRM STORBUF controls the VM Guest Users not to be sent to the eligible list in spite of memory over commitment.

Default value for SRM STORBUF is : Q1=125% Q2=105% Q3=95%. Usually Users are moved to 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%, Q3=ccc%, users are moved to Dispatch list from Eligible list with the following conditions.

- User is E1 : $\text{Total WSS} + \text{Target E1 User's WSS} \leq \text{DPA} \times \text{aaa}\%$
- User is E2 : $\text{Total WSS} + \text{Target E2 User's WSS} \leq \text{DPA} \times \text{bbb}\%$
- User is E3 : $\text{Total WSS} + \text{Target E3 User's WSS} \leq \text{DPA} \times \text{ccc}\%$

where WSS means Working sets and DPA means dynamic paging area.

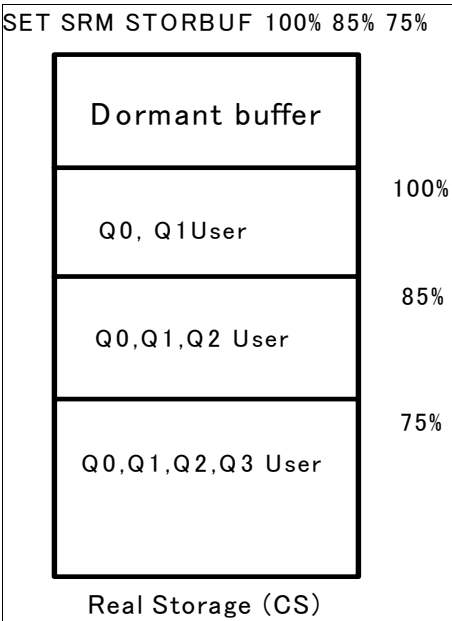


Figure 7-2 example SET SRM STORBUF

But for larger production Linux guests or servers, it is recommended to set 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 Eligible list, instead allowing 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

you will get done.

Example 7-3 Default SRM LDUBUF values.

```
LDUBUF : Q1=100% Q2=75% Q3=60%
```


In the Example 7-3, 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 ever be able to use only 60% paging "capacity" with our Linux guests. So we should raise the LDUBUF settings. An example would be

Example 7-4 example SRM LDUBUF setup for Linux guests

```
SET SRM LDUBUF 100 100 100
```

which would mean all guests in any queue would all be able to compete for 100% of your paging "capacity".

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 and 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 TCPIP, and your more important production linux servers should have this option.

Points to note :

- Userid becomes transaction class 0 and never waits in the eligible list.
- Impacts decision to move from E-list to D-list on userid basis.
- Used for key server virtual machine (process synchronous requests from end users) and virtual machines that demand the best performance.

Example 7-5

```
SET QUICKDSP userid ON
```

The Quickdsp setting impacts a single userid. It should be used for any server virtual machine that is critical to end user response time or holds key system resources. Over use 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.

To 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 to protect 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 application(s) 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-2GB 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-2GB storage pretty hard. CP prefers to resolve below-2GB contention by using XSTORE. Consider configuring at least 2 GB to 3 GB of XSTORE so as to back the below-2GB central storage, even if central storage is otherwise large.

7.3.1 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 if any virtual machines can be made smaller.



8

Hardware related problems

This chapter introduces basic configuration of hardware, device recognition and management in Linux on System z. Describes hardware problem determination basic, and some case studies.

8.1 Basic Configuration of Hardware

This subchapter introduces typical one of the hardware used by Linux on System z. If you encounter a hardware problem, these diagram help for understanding the hardware configuration and connectivity.

Figure 8-1 on page 238 is the basic configuration for this chapter.

- ▶ Processor : IFL, CP
- ▶ Channel Adapter : ESCON®, FICON, OSA-Express
- ▶ I/O Devices : Disk, Tape, Terminal, Director

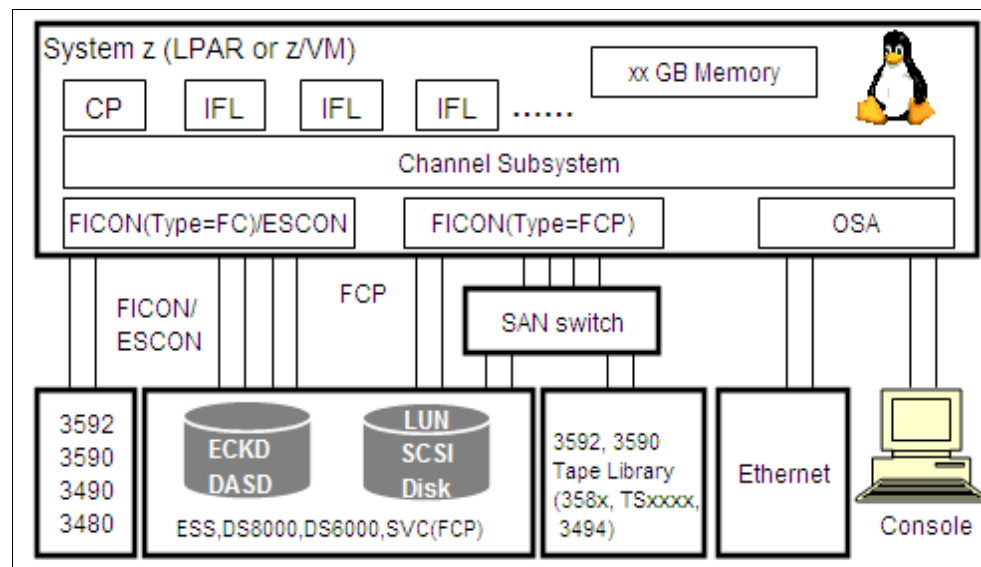


Figure 8-1 Basic hardware configuration for Linux on System z

8.2 Device recognition and management

This subchapter describes method of recognizing devices of Linux(device file, sysfs, udev, device driver, common I/O, qdio), and of cofirming device status.

8.2.1 Whole image of device control

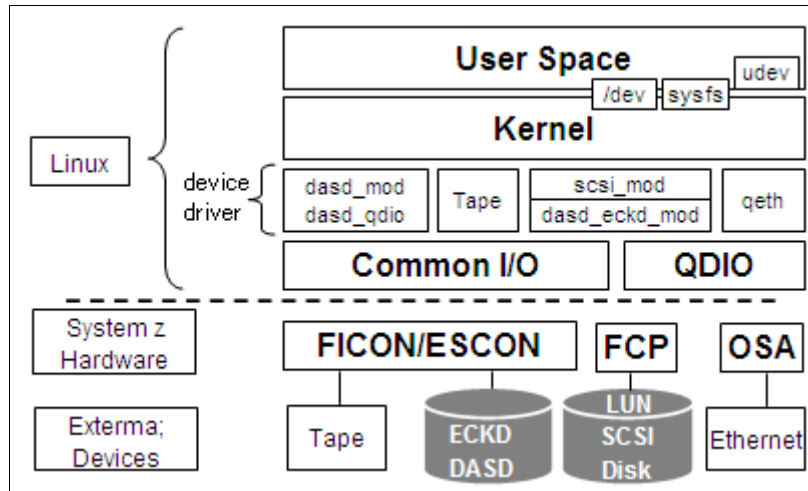


Figure 8-2

- ▶ **Device Driver**
 - Drivers that are often used in Linux on System z.
 - DASD(`dasd_mod`, `dasd_eckd_mod`, `dasd_fba_mod`)
 - TAPE(`tape`, `tape_34xx`.)
 - FCP/SCSI(`zfcp`, `scsi_mod`, `st`, `sg`, `IBMtape`)
 - Network(`qeth`)
- ▶ **Common I/O Layer**
 - A layer that executes common I/O operations that are not depend on devices.
 - Actions in CommonI/O layer
 - recognition of channel path
 - detection of devices
 - recognize device type and control unit type

- qdio driver
 - It controls I/O of devices which use qdio mode

8.2.2 Kernel Layer

Kernel recognizes and manages hardware with device files, sysfs, and udev.

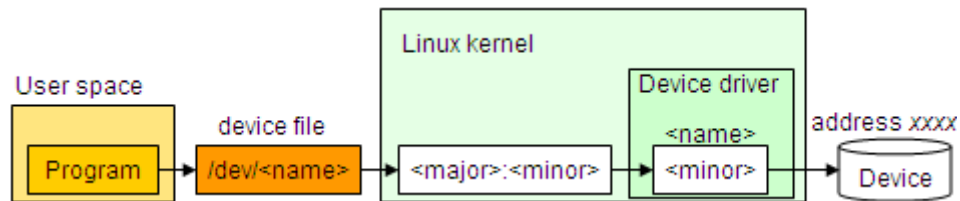


Figure 8-3

- What is device file?
 - Linux treats a device as a file
 - The use program control devices using device files
- *** Kernel manages devices using major number and minor number
 - major number : number which indicates device driver ex) 94 -> DASD
 - minor number : number which indicates each device
- Making device files
 - some device files are created automatically when Linux is installed
 - Linux creates device files that are often used in advance
 - udev
 - mechanism which creates device files automatically
 - it is called by hotplug, and registers a device into sysfs
 - create manually (in case of adding new device, in case of not being created automatically by udev)
 - create by mknod command

udev

udev is a system that dynamically creates and deletes device file. (adopted in kernel2.6)

By creating and deleting dynamically device files, device files of only devices which are actually connected to system are created under /dev directory, and it is easy to confirm that what devices are connected now and whether devices are connected correctly.

However, in SLES, necessary minimum device files always exists in the system and these files are not deleted dynamically.

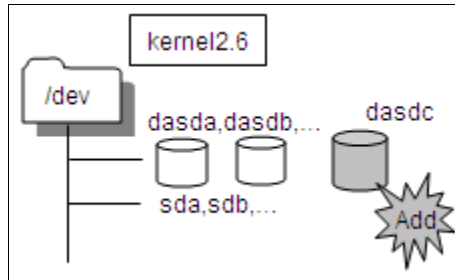


Figure 8-4

Mechanism of device recognition by udev (SLES9)

1. New device is added and machine check occurs
2. Machine check handler inform CIO layer of it
3. Information about added device is acquired in Common I/O. a new entry is created in sysfs
4. Hotplug is called
5. Udev is called
6. Udev references sysfs
7. Device file is created

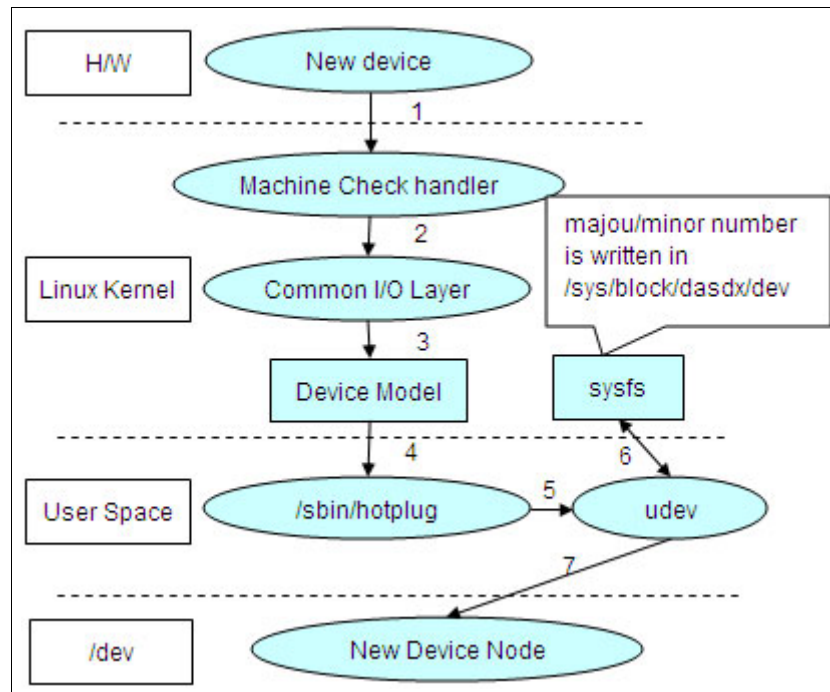


Figure 8-5

sysfs

- ▶ Sysfs offers an interface for user space to access to data structure in kernel space
 - Dynamic device adding/deleting
 - by hotplug agent
 - by user operations
 - option setting of device and driver (log level, etc...)
 - confirmation of device status (online/offline, etc...)
- ▶ Example of configuring 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 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 Driver

- ▶ DASD device driver
  - dasd\_mod, dasd\_eckd\_mod, dasd\_fba\_mod
  - device driver which 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/dasdx`
    - first partition is `/dev/dasdx1`, second partition is `/dev/dasdx2`
    - the maximum number of partitions is 3
- ▶ zfc device driver
  - device driver which is used to access fibre-channel-attached SCSI devices
  - Channel subsystem is not used, and buffer I/O of QDIO is used
    - Linux has to process path control and error recovery by itself because channel subsystem is not used.
  - The information about HBA and LUN in storage server side is needed to recognize a LUN
  - Linux controls recognized LUN by using device file `/dev/sdx`
- ▶ Network device driver
  - Network device is controlled by network device driver
    - Device file and major/minor number are not created different from disk and tape
    - Interface is created and related to a device
  - Default name of interface depends on type of interface
    - ex) Ethernet(`eth0`,`eth1`,`Ac`), HiperSockets(`hsi0`,`hsi1`,`Ac`)
    - The name of interface can be changed

<qeth>

- Device driver which is used to access OSA-Express of QDIO mode and HiperSockets
- Channel subsystem is not used and buffer I/O of QDIO is used
- qeth device driver needs 3 subchannel to recognize OSA device
  - control read : must be even
  - control write : must be the device bus-ID of the read subchannel plus one
  - data : can be any free device bus-ID on the same CHPID.

## 8.2.4 Common I/O

I/O processing that doesn't depend on the device is executed.

- ▶ Starting I/O
  - It demands to start I/O processing to channel subsystem.
- ▶ Completing I/O
  - Receiving I/O interruption from Hardware, when I/O is completed.
  - The subroutine of device driver corresponding to H/W is called.
- ▶ Error handling at channel and sub-channel
  - When the H/W event is occurred, the machine check interrupt is received.
  - The recovery processing according to the event is executed.
- ▶ Recognizing the devices that the channel subsystem manages.
  - Acquiring sub-channel information
  - Acquiring CU/device type/Model Number
  - Inspection of channel paths of each device

### Overview

I/O request: Creating CCW, and execute SSCH (Start Sub Channel) instruction.

I/O completed: Notified to operating system by I/O interruption. the status is stored to IRB.

Machine check interruption: Hardware events are notified by CRW.

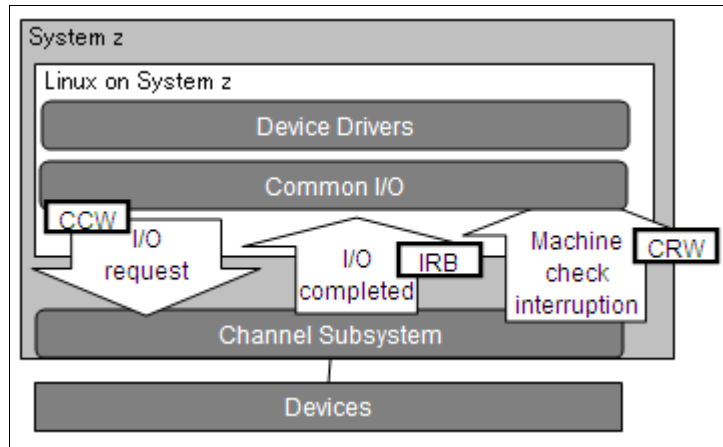


Figure 8-6

\*\*\*IRB (Interrupt Response Block) : Storing device and channel status.

\*\*\*CRW (Channel Report Word) : Storing event information. (ex. channel link down/link up)

### CCW (Channel Command Word)

I/O requests to devices (hardware understandable instructions). These Information included in CCW.

- Hardware instruction such as read or write.
- Data or buffer address on memory.
- Number of bytes for input/output data

The kind of the instruction is different in each device.

### CRW(Channel Report Word) Messages

Notifying the status of the I/O operation, the device, and the channel paths. For example, link up/link down, attaching devices. Example of message when 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)

---

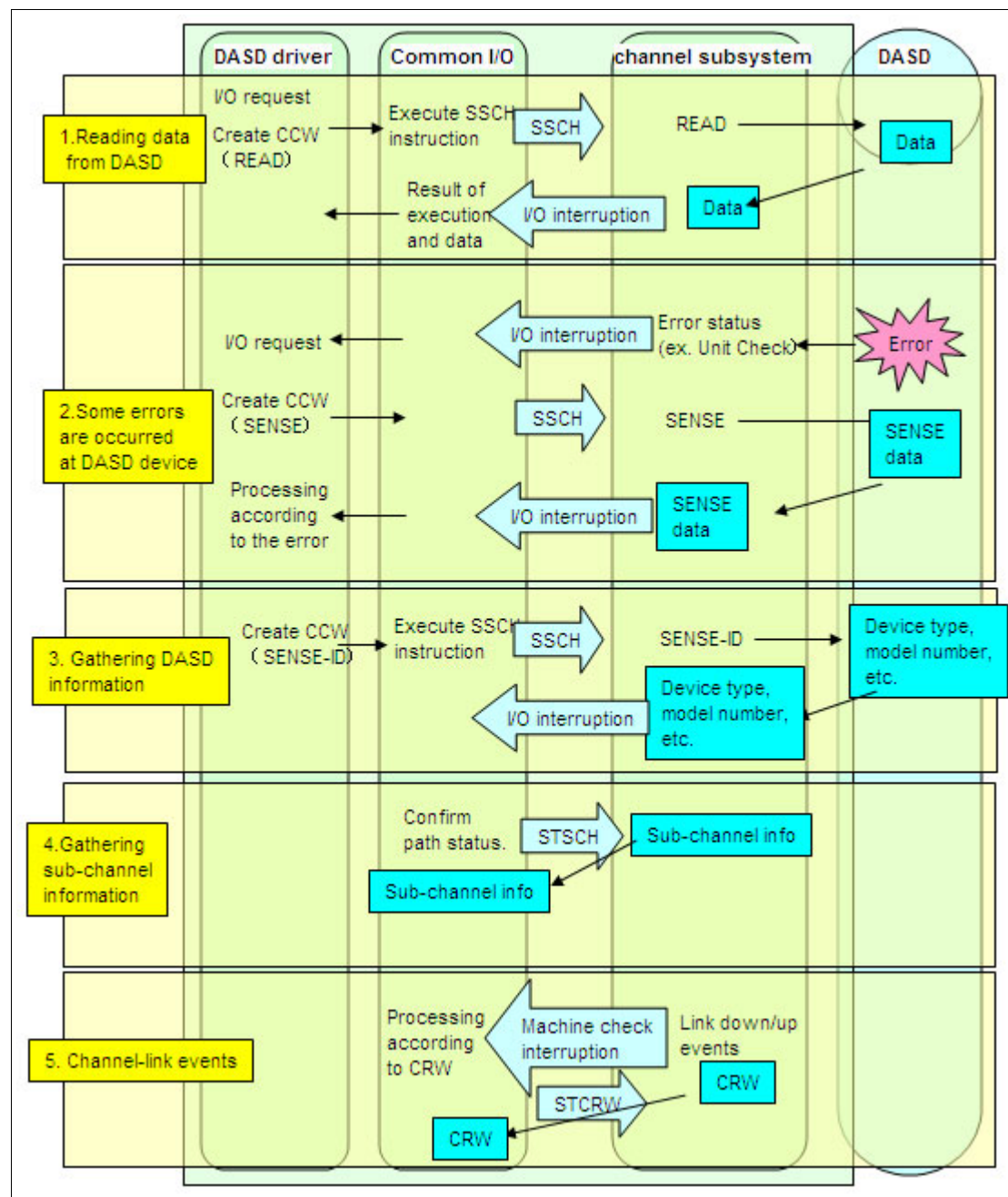


Figure 8-7 Outline of I/O Processing

8.2.5 QDIO

QDIO = Queued Direct I/O

Architecture that provides efficient data transfer. Direct communication between memory and adapter devices(DMA: Direct Memory Access). I/O processing paths are shortened.

8.2.6 Confirming Status of Channel-path and Devices

Commands files of status confirmation of devices. It is possible to confirm devices status provided by sysfs. It is important to confirm the normal status in normal circumstances to finding abnormal status

Table 8-1

|         | Commands/Files                            | Description and Contents                                                      |
|---------|-------------------------------------------|-------------------------------------------------------------------------------|
| General | lscss                                     | The devices status that Common I/O manages are displayed.                     |
|         | /etc/sysconfig/hardware/*                 | They are including device information.                                        |
| DASD    | lsdasd                                    | Composition information on DASD is displayed.                                 |
|         | /sys/bus/ccw/drivers/dasd-eckd/0.0.<addr> | Composition information on DASD is included.                                  |
|         | /proc/dasd                                | Composition information and the statistical information of DASD are included. |
| FCP     | lsscsi                                    | Information on SCSI devices are displayed.                                    |
|         | /sys/bus/ccw/drivers/zfcp/0.0.<addr>      | They are including composition information of SCSI devices.                   |
|         | /proc/scsi/IBMtape, /proc/scsi/IBMchanger | They are contain FCP tape devices information.                                |
| Network | lsqeth                                    | Composition information on the qeth network devices are displayed.            |
|         | /sys/bus/ccwgroup/drivers/qeth/0.0.<addr> | They contain qeth network devices information.                                |
|         | /proc/qeth                                | It contains qeth network devices information.                                 |

Confirming Status of Channel-path and Devices

Are the channel and the device correctly recognized ?

- It is possible to confirm by lscss command



*Example 8-2*

| 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

### Physical Path Status

- PIM/PAM/POM
  - A part of path status information which are included in Subchannel Information Block
  - Get them by ÅgSTORE SUBCHANNELÅh instruction.
- It is possible to confirm them by lscss command

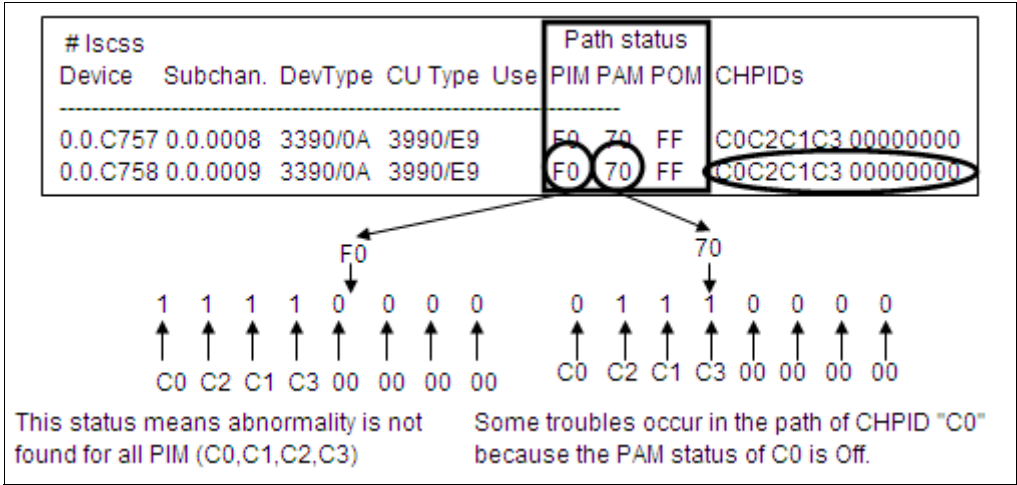


Figure 8-8

Confirming device status

- ▶ DASD status
  - lsasd command
  - /proc/dasd/devices

Example 8-3

```
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
```

- ▶ FCP device status
  - lsscsi command

Example 8-4

```
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]
```

- ▶ OSA Status
  - lsqeth command (only for SLES)

*Example 8-5*


---

|                   |                   |
|-------------------|-------------------|
| 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 describes

- ▶ Explaining check points to divide hardware and software problem.
- ▶ It explains the procedure to divide each step of the problem of the hardware recognition and to distinguish the problem.

### 8.3.1 Initial determination of hardware or software problem

#### Confirmation phenomenon

If problem happens, confirm phenomenon at first. Don't forget gathering first material (dbginfo.sh, s390dbf). And confirming the presence of changing system.

- Devices can't be recognized
- The performance is not an expectation
- Devices cannot be used

- Though there are no problem in device recognition and performance, some error are reported
- etc

After confirming phenomenon, decide by the component that can be obviously judged.(storage subsystem trouble? network trouble?) And judge whether it is a trouble related to I/O. I/O trouble needs to be investigated finely.

H/W trouble shows some messages.

- RSF(Remote Support Facility) notification
- The icon status of HMC
- Check stop of LPAR

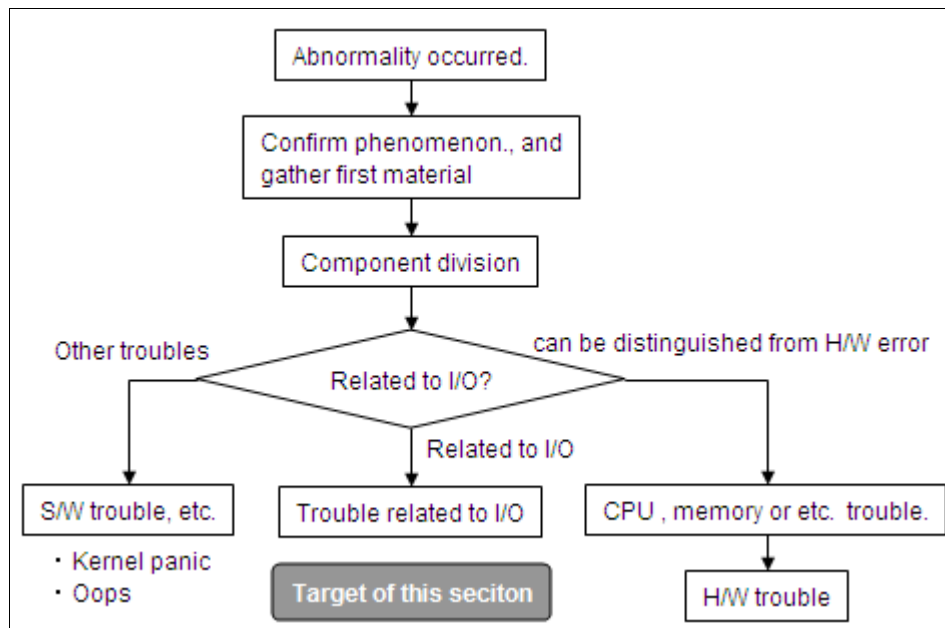


Figure 8-9

### Distinct H/W or S/W problem

For distinction the problem is related on H/W or S/W, error messages and problem scope(single system or multi system) are good hint. Hardware error messages are shown Linux logs, HMC, z/VM console, or z/VM EREP user.

/var/log/messages on Linux doesn't have the messages which is shown during Linux hang, down, or reboot.

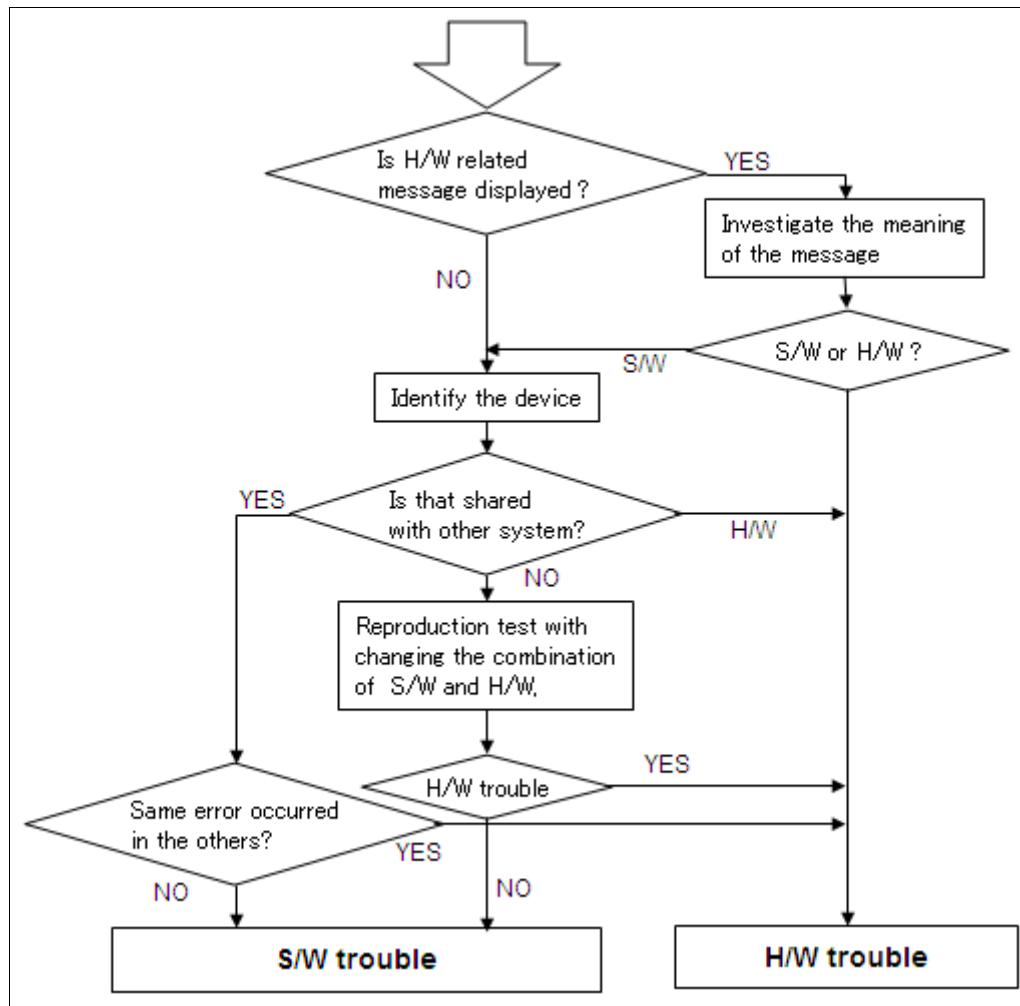


Figure 8-10

### 8.3.2 Investigation into the H/W related messages

H/W error messages in z/VM basically have error code and its meaning. Check message and code manual of each component and each version of z/VM.

We have no manual about H/W messages in Linux. Searching the meaning of messages from the past cases is good approach for investigation, in Technical Q&A library, Bugzilla, Technical Q&A library. Or, Linux source code have its meaning.

This section has these sample messages for hint.

- Example of H/W error messages of DASD (1)
- Example of H/W error messages of DASD (2)
- Example of error messages of FCP devices
- Example of the messages which look like H/W error messages (1)
- Example of the messages which look like H/W error messages (2)

### **Example of H/W error messages of DASD (1)**

Case: sense information is displayed in the messages

- ▶ We cannot know the bit of sense data from the messages
  - “Environmental data present” (sense data byte 2 : 10)
  - “FORMAT F - Operation Terminated” (sense data byte 7 : F0)
    - > Cache Subsystem Error Report
  - “waiting for state change pending interrupt” (sense data byte 25 : 1D)
    - > State-Change Pending = “Long-Busy”
- ▶ We can get a point through investigation into source codes
  - That is the information to send to a H/W engineer

*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
```

---

### **Example of H/W error messages of DASD (2)**

Case: example of “Equipment Check”

- ▶ Equipment Check (EQC) usually indicates H/W error
- ▶ The message below indicates EQC of Reset Event
  - Reset Event is returned from control units on the timing of the first access after channel path reset. Channel path is reset when the reconnection is

occurred after the physical connection of ESCON or FICON is disconnected.

- We cannot find that the connection is disconnected 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 of FCP devices

Case: Error of the tape drive with FCP connection

The following messages output during backup to tape. Check the address of the adapter from the error messages at first.

1. Discover Address (ADISC) Request failed
2. The port is not connected with SAN network
3. Recovery failed.

*Example 8-8 /var/log/messages*

---

```
Apr 3 07:07:21 xxxxxxxx kernel: zfcpx: 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: zfcpx: The adapter 0.0.1902 reported the following
characteristics:
Apr 3 07:07:52 xxxxxxxx kernel: WWNN 0x5005076400cd191f, WWPNN 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: zfcpx: 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 = Extended Link Service. In this case, ELS requests the address information of the target port.

\*\* gid\_pn = Get Port Identifier - Port Name: send the port name to name server and get the port ID.

### Example of the messages which look like H/W error (1)

Case: In case of trying to recognize the DASD which is not formatted.

The following messages is displayed, but it doesn't mean H/W error. (The error messages will not be displayed if the DASD is 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 of the messages which look like H/W error (2)

The following messages is displayed when ext3 filesystem is used. The transaction barrier function is not supported by the storage systems.

- Transaction barrier function : a supplementary mechanism for keeping data integrity.
- How to avoid
  - Add the parameter “barrier=off” to ziopl.conf

*Example 8-10 /var/log/messages*

---

```
JBD: barrier-based sync failed on dasdxx - disabling barriers
```

---

## 8.3.3 Approach for H/W problem

### Contact to H/W support person

We need H/W support person to investigate into H/W problem.

- Basic informations.



- Machine type
- Serial number
- Telephone number of the Machine room
- ▶ The necessary informations.
  - What happened
  - What were you doing when that happened
  - When did it occur
  - Did you change anything of the setting just before that happened
  - The information to specify the H/W
  - You should gather the information about the H/W(ex. address) beforehand
  - Error messages on Linux
  - The information which 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.

- The performance trouble about Disk I/O which are on different Storage systems.
- The network connection trouble in the environment which has plural NICs.

Refer to storage, network and performance determination chapter for detail.

After specifying the device in error, you should check the status and distinguish the error. (H/W or S/W)

### **Check the status of DASD recognition**

#### **1. Common I/O**

- Check Iscss output
  - Check whether the address of the target device is displayed
  - Check whether “DevType” and “CU Type” are collect

#### **2. 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`
3. Online status
    - Check `lsdasd`
    - Check `lscss`
    - Check `/sys/bus/ccw/drivers/dasd-eckd/0.0.<address>/online`

## Check the status of FCP device recognition

1. FCP port on System z
  - Check the `lscss`
    - Use="yes" means FCP port is online on System z.
  - `/sys/bus/ccw/drivers/zfcp/0.0.xxxx/online` means the same.
2. FCP port on storage system
  - Check `/sys/bus/ccw/drivers/zfcp/0.0.xxxx/` directory.
    - There is a directory which is named with WWPN, FCP port is online on storage system.
3. LUN recognition
  - `/sys/bus/ccw/drivers/zfcp/0.0.xxxx/yyy...yyy/` directory (yyy...yyy is WWPN)
    - There is a directory which is named with LUN number, LUN is recognized.
4. SCSI disk recognition
  - Check `lsscsi`
  - Check `/proc/scsi/scsi`

## Check the status of OSA recognition

1. Common I/O
  - Check `lscss`
2. Device driver
  - Check symbolic link about devices on `/sys/bus/ccwgroup/drivers/qeth/*`
  - Check entry on `/proc/qeth`
3. Online status
  - Check `lscss`
  - Check status on `/proc/qeth`

## Investigation of sharing devices

Basic Linux on System z environment, system resources(CPU, Memory, Channel, Network) are shared. When H/W trouble happens, same problem will occurs on all systems which share the H/W.

If it's difficult to distinct the trouble is related on H/W or S/W, additional test with changing S/W version or combination of H/W and S/W.

- ▶ Change S/W version
  - If the trouble doesn't happen when S/W version is changed.
    - S/W problem
- ▶ Assign same H/W to the other system.
  - The trouble happens on the other system.
    - H/W problem
  - The trouble doesn't happens on the other system.
    - S/W problem
- ▶ Assign different H/W to the same system.
  - The trouble happens on the other H/W.
    - S/W problem
  - The trouble doesn't happen on the other H/W.
    - H/W problem.

### 8.3.4 Actions after dintinction with S/W or H/W problem

It seems S/W problem.

- Determine on device driver layer.
- Determine on each component.

It seems H/W problem.

- Contact to H/W support person.
- Preparation the procedure for changing H/W exchange.

If we cannot distinct S/W or H/W problem.

- Test the problem reproduction, and gather informations.
  - Confirm the problem reproduction procedure.
  - Set the propper debug level in s390dbf and other log and trace.
  - Run problem reproduction test

- Get informations with dbginfo.sh or other tools.

## 8.4 Case Study

This section introduces some H/W trouble cases.

### 8.4.1 Memory card trouble

#### State

- ▶ Memory error messages are displayed on HMC.
- ▶ Notice by RSF
- ▶ Check stop or system hung is occurred in some LPARs which share the same memory card.

#### Environment

- ▶ RHEL4 update 4 on System z9® LPAR

#### Gathered informations

- ▶ System messages on HMC OS messages

#### Analysis step

- ▶ Contact to H/W support person.

#### Result and solution

- ▶ H/W problem
  - Exchange memory card solves the problem.

#### Tips

- ▶ If RSF is reported, we can regard the trouble is H/W problem.

### 8.4.2 OSA trouble

#### State

- ▶ Network interface goes down with qeth error meesages

*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 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)

## Gathered informations

- ▶ 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 H/W problem happens.
- ▶ Contact to H/W support person

*Example 8-12 /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 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.

```

---

## Result and solution

- ▶ H/W problem
  - Exchange OSA card solves the problem.

## Tips

- ▶ Device status or subchannel code often show hints for distinction H/W or S/W problem.

## 8.4.3 DASD error

### State

- ▶ DASD error messages in /var/log/messages

*Example 8-13 /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)

### Gathered informations

- ▶ dbginfo.sh (especially /var/log/messages)

### Analysis step

- ▶ Search the conditions of the error messages.
  - This messages are error when I/O requests are started.
- ▶ Error messages are displayed in some device address on same path.
  - Confirm the H/W error or change with H/W support person.

### Result and solution

- ▶ H/W temporary error
  - ESS detected the temporary error of FC adapter at the same time.
  - ESS do warm start (about 2 seconds).
  - There was no problem because ESS recover normally.

## Tips

- ▶ It seems path error if error occurred in some address on same path.

## 8.4.4 Network interface error in reboot

### State

- ▶ “No interface found” message of network interface is displayed in boot.msg after reboot.
- ▶ Bonding interface including the interface is not configured.
- ▶ Same phenomenon is occurred in some LPARs share OSA port.

*Example 8-14 /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)

### Gathered informations

- ▶ dbginfo.sh

### Analysis step

- ▶ Same phenomenon is occurred in some LPARs share OSA port, so it seems to be H/W trouble.
- ▶ Confirm with H/W support engineer, but receive an answer that H/W status is normal.
- ▶ Found failure about the device from the lscss
  - “DevType”, “CU Type” is received as response of senseID from H/W
- ▶ Realize that the cause is either one of the following
  - SenseID isn’t send to the device.
  - SenseID is send, but OSA doesn’t reply.
  - Check the process of device initialization from source code and trace. (by support center)
    - SenseID is send when LACP turn ON.

- Try to get the debug trace in customer environment
    - i. set debug level
 

```
echo 6 > /proc/s390dbf/cio_msg/level
```

```
echo 6 > /proc/s390dbf/cio_trace/level
```
    - ii. LCHP OFF/ON
 

```
echo "offline" > /sys/devices/css0/chp0.63/status
```

```
echo "online" > /sys/devices/css0/chp0.63/status
```
    - iii. get dbginfo
 

```
*** After LCHP OFF, status doesn't change to ONLINE
```
  - Ask to search about H/W again.
  - Forcibly obtained the detail information of OSA (Forcelog \*)
    - Check stop happens.
- \* Forcelog: error logs obtained by operating SE.

## Result and solution

- ▶ H/W problem
  - Exchange OSA card solves the problem.

## Tips

- ▶ Same phenomenon is occurred in some LPARs share OSA port.
  - It seems more likely that of H/W trouble.
- ▶ When it is difficult to determine H/W trouble or S/W trouble, reproduce the problem and get s390dbf.

## 8.4.5 DASD recognition trouble in boot process

### State

- ▶ System doesn't recognize the DASD in reboot.

*Example 8-15 /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 vgHEPRRC01arctest0401.
/dev/dasdca1: lseek 0 failed: invalid argument
/dev/dasdca1: lseek 0 failed: invalid argument

```

---



## Environment

- ▶ SLES9 on System z

## Gathered informations

- ▶ /var/log/boog.msg
- ▶ dbginfo.sh (includes /var/log/messages)
- ▶ siga no
- ▶ Stand Alone Dump

## Analysis step

- ▶ DASD is recognized as non-label according to /var/boot.msg.
- ▶ Size of DASD partition's size is abnormal on /proc/partitions in dbginfo.

*Example 8-16 /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:V0L1/ 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 8-17 /proc/partitions*

---

```
94 149 7212144 dasda11
94 312 7212240 dasdca
94 313 9223372036854775796 dasdca1
94 244 7212240 dasdbj
```

---

- ▶ [ search by support center and developer ]
  - Search the flow of I/O process from s390dbf in dump.
  - The function for recognition DASD doesn't get the volume label information and partition information.

## Result and solution

- ▶ S/W problem
  - It's same problem as reported one in SLES10.
  - Release patch for current kernel of SLES9.

## Tips

- ▶ Determine this trouble is same as one of SLES10 according to /var/log/boot.msg, dbginfo and dump information comprehensively.



# 9



## Installation and Setup Problems

This chapter is a guide to help you to identify most problems you could encounter with 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
- ▶ Gather information regarding installation
- ▶ Scenarios

## 9.1 Understanding the installation process.

First of all, before start a Linux installation we need to check for some system requirements (e.g memory, network , disk space). To know in details which are these requirements go to your preferred distributor linux site.

<http://www.novell.com>

<http://www.redhat.com>

Linux installation on System z can be performed in 2 different scenarios:

**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 that linux use a part of physical memory on system.

It is possible to install Linux on System z either from graphical or from 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. Its required also another server running.

► **X11 Forwarding.**

From your linux machine, open a command prompt and type the following:

```
$ ssh -X servername.company
```

where:

**-X** - This option enables X11 forwarding. This can also be specified on a per-host basis in a configuration file.

**servername.company** - Its a server machine where the linux will be installed. (e.g lnx.itso.ibm.com).

► **VNC.**

From your linux or windows machine , we can use VNC Client to access the server and install linux.

**Note:** In most of cases VNC option is the best solution.

There are at least 4 methods installations available to install your linux distribution: Install from a Hard Disk or a DASD, Install via NFS, Install via FTP and Install via HTTP.

When we are using NFS, FTP or HTTP methods, we need to fill an additional screen configuration.

**Note:** For SuSE SLES 10 and RHEL 5.1 we can choice Layer2 connection for connect to NFS and FTP server.

When we start the installation, we will decide configurations about language selection until packages available to be installed. Bellow, steps that we will navigate during the installation process:

**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 be supported you must configure in the language support configuration panel

**Disk Partitioning Setup .** At this option, we can choose into Automatic Partitioning and Manual disk partitioning. Here, we will configure the partitions sizes used by linux. See example at Figure 9-1

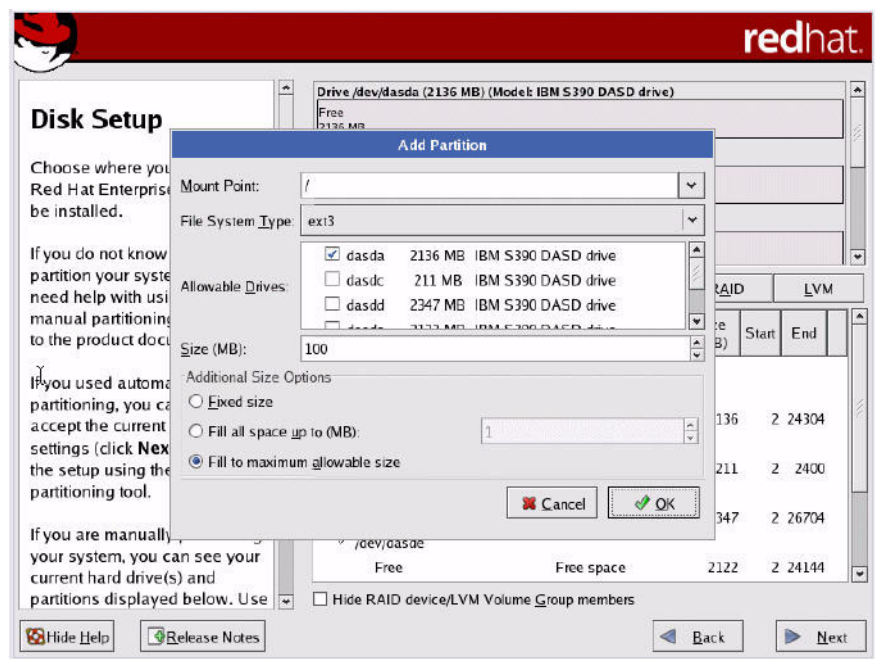


Figure 9-1 Adding partitions on Linux Installation

**Note:** For a production server is highly recommended uses the option Manual disk partitioning

**Network configuration.** Here, we will configure hostname, gateway and DNS. If we have multiple network cards, each device have its own configuration screen. See example at Figure 9-2

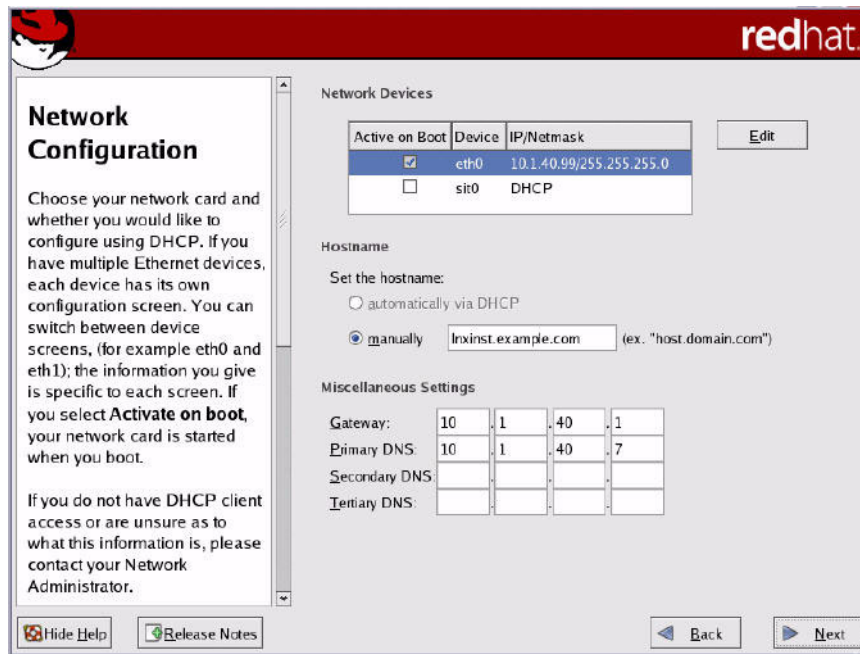


Figure 9-2 Configuring network

**Firewall configuration.** At this point, we can decide to prevent access to your computer. We can choose into No Firewall and Enable Firewall option. If Enable firewall is checked we can allow access to some services as Remote Login (SSH), File transfer (FTP), Mail Server (SMTP) and Web Server (HTTP/HTTPs). See example at Figure 9-3 on page 272

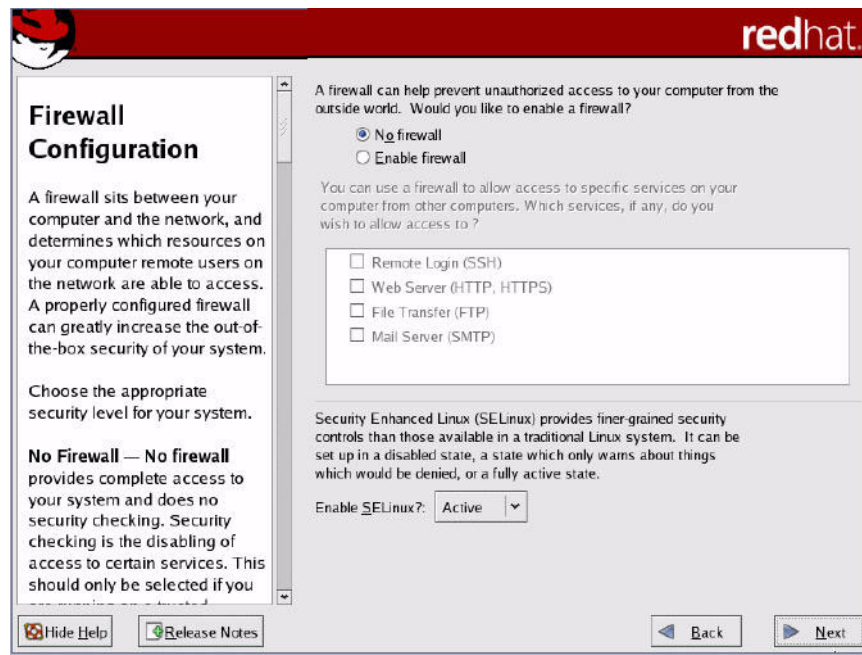


Figure 9-3 Firewall configuration

**Note:** If you prefer, you can enable/disable this services after the system is installed.

On following screens we will have a possibility to select a language suport, set the timezone, root password and select packages we want to install.



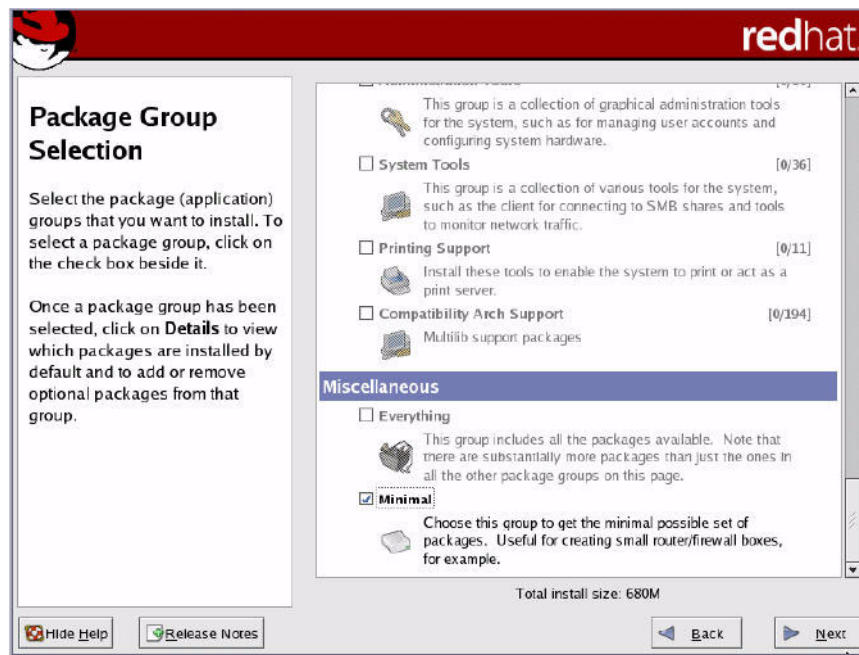


Figure 9-4 Package configuration.

After this, Linux will be installed and ready to use..

**Note:** For more information about the Linux detailed installation on S/390 architecture can be found in IBM z/VM and Linux on IBM System z - Virtualization Cookbook for Red Hat Enterprise Server, SG24-7272-00 and in the site <http://www.ibm.com/linux>

## 9.2 Gathering information regarding installation process

**Note:** This chapter was written based on Red Hat Enterprise Linux version. There are comments included when the same topic have a different file name or location for SUSE SLES

## Gathering information provided during the installation

During the installation, all informations are saved in the file `/root/install.log`, but we are available to access this file once we reboot the system.

At this file we will 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 at `/var/log` directory. At this directory we can find system and applications default logs. An example of application logs We can see the `gdm` and `cup` directories that contain log for the applications **G**nome **D**isplay **M**anager and **C**ommom **U**nix **P**rinting **S**ystem respectively. In the figure below, we can verify more logs available at `/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 to gather information , the daemon **syslogd** and the file **syslog.conf** are essential keys to understand what is printed out to `/var/log/` files.

**Syslogd** is a daemon that read and log 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 and where the information will be available.

In the example below shows a `syslog.conf` file. Note that all information regarding notice, kern debug, etc. will be recorded on `/var/log/messages`. See Example 9-2

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 what kind of information we want to see on /var/log/messages. The file syslog-ng.conf is located at /etc/syslog-ng directory

Analyzing syslog.conf example, We can note that:

The line *mail.\* /var/log/maillog* is responsible for writing all information regarding mail on 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 allow 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

```

---

To a better understanding we will comment some parts of `/var/log/messages` logs according to the topics below.

### ***Informations regarding applications (e.g. SuSEfirewall)***

This part of the log, shows information about the SuSEfirewall application printed. We can note that there is information about firewall rules status process, warnings, etc. All information printed here is a execution result from 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
```

---

**Informations regarding linux kernel**

This part of the log, shows information about Kernel printed. These pieces of information are 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
```

---

### ***Informations regarding network cards.***

This part of the log, show information regarding network cards printed. These pieces of information are recorded while the network card is starting. If we have more then 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 will 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 2 possibilities:

- Problems during the installation

Are problems that occur while running program interface installer.

As example : problem to create partitions, packages dependencies, devices not found, also problems with network setup.

- Problems after the installation

Are problems that occur after Linux install is done and successfully.

As example : devices configurations, problems to log in linux, etc.

### 9.3.1 Trouble during the installation

#### Scenario 01: Problems to clone 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 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
```

*/etc/fstab example.*

## 9.3.2 Trouble after the installation

### Scenario 01: Forgotten root password (SuSE or RedHat)

**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 we typed wrong during the installation.

**Solution:** To solve this problem, we will 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. Here, we can reboot your system using init 6 or typing the command below:

```
shutdown -r now
```



# Booting Problems

This chapter is a guide to help you to identify most problems 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:

- ▶ Understanding the boot process
- ▶ Gather information regarding boot
- ▶ Commands
- ▶ Boot Problem Examples
- ▶ Scenarios

## 10.1 Understanding the boot process.

The Linux Operating System for System z has 4 phases on its boot process. They are: IPL, Boot Loader, Kernel and Init phases. See Figure 10-1

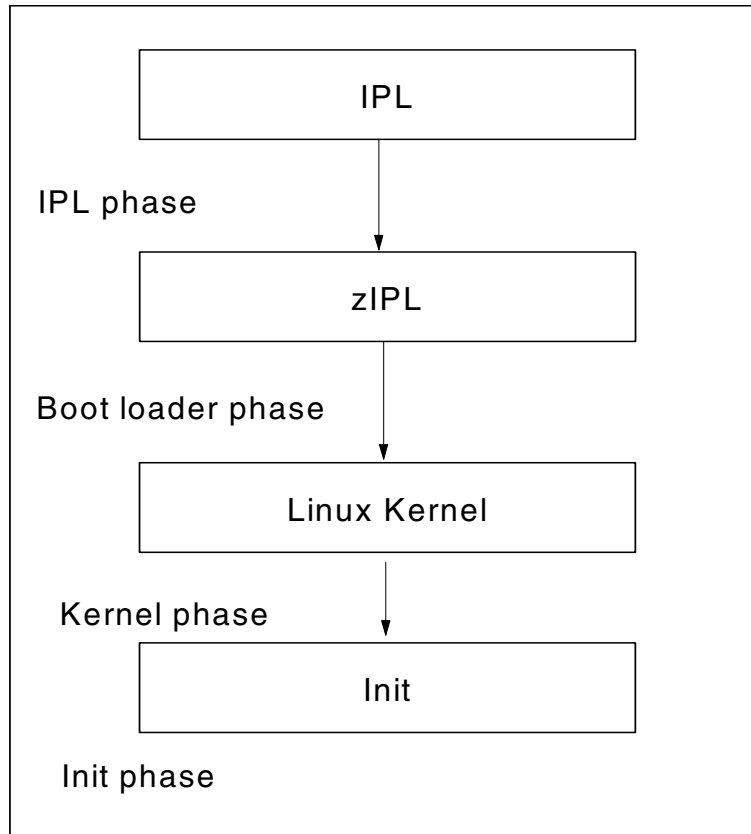


Figure 10-1 boot process.

Now, we will 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 like boot is for other architectures. See Figure 10-2

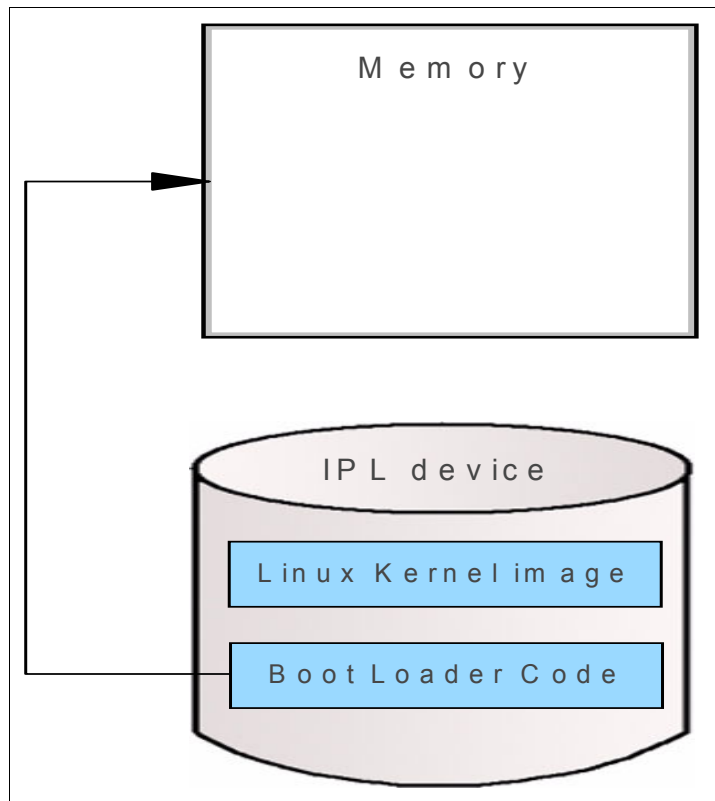


Figure 10-2 IPL phase.

IPL phase, consist to read the boot loader code from operator-specified device into memory and start boot loader (ziPL).

DASD, SCSI disk, Tape and VM reader. can be used as IPL device.

### 10.1.2 Boot Loader phase

The main bootloader function is to load other software/program for the operating system to start. It's found at the start of the IPL device. The bootloader used for IBM S/390 and zSeries machine is called ziPL. Boot loader code is made by /sbin/zipl command and installed on user specified IPL device. Figure 10-3 shows an example of how ziPL works.

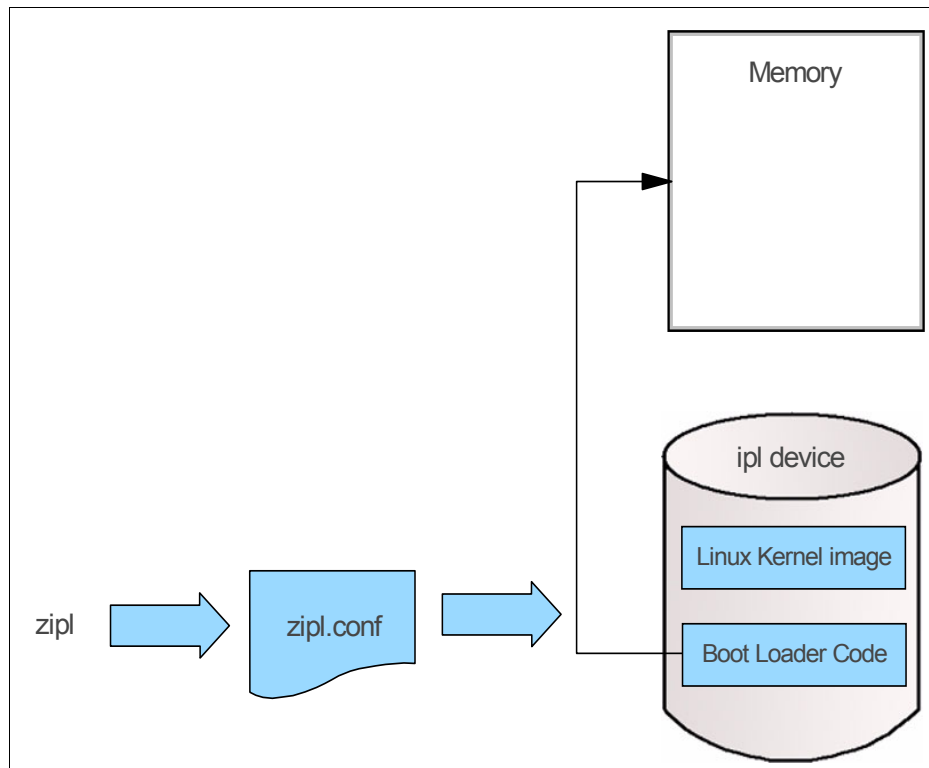


Figure 10-3 How zipl works.

Bootloader phase consist in Load Linux and Initial RAM disk image (initrd) and start linux kernel, then boot the Linux Operating System.

After loading boot Linux Kernel, the control is passed to the Linux Kernel. As showed in Figure 10-4:

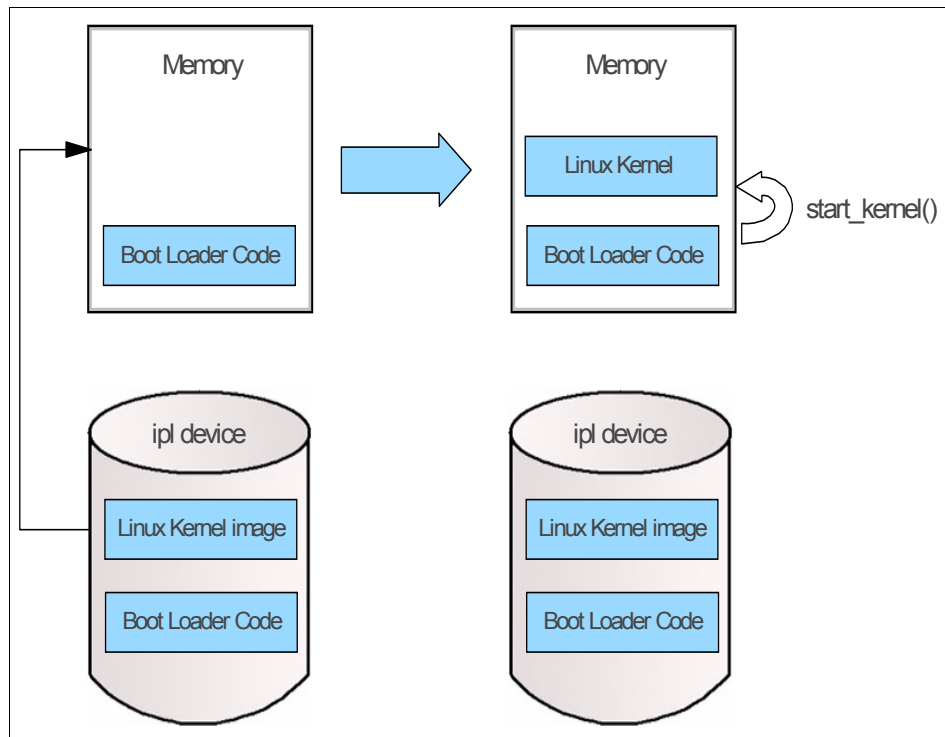


Figure 10-4 Boot Loader phase

You will find a configuration for a zIPL. See Example 10-1

#### Example 10-1

```
[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.750000000BALB1.89e9.10-part1
elevator=deadline TERM=dumb"
```

---

### 10.1.3 Kernel phase

Kernel is the heart of operating system. Some of kernel functions are manager system resources (memory, CPU.etc) and make the communication between software and hardware, etc.

The kernel phase consist in probe and initialize the hardware, so initialize kernel subsystems and decompressing initrd and load some devices drivers. After that, mount root file system with read only access and check the filesystem with fsck program, then start /sbin/init. After this kernel will sleep. As showed at Figure 10-5

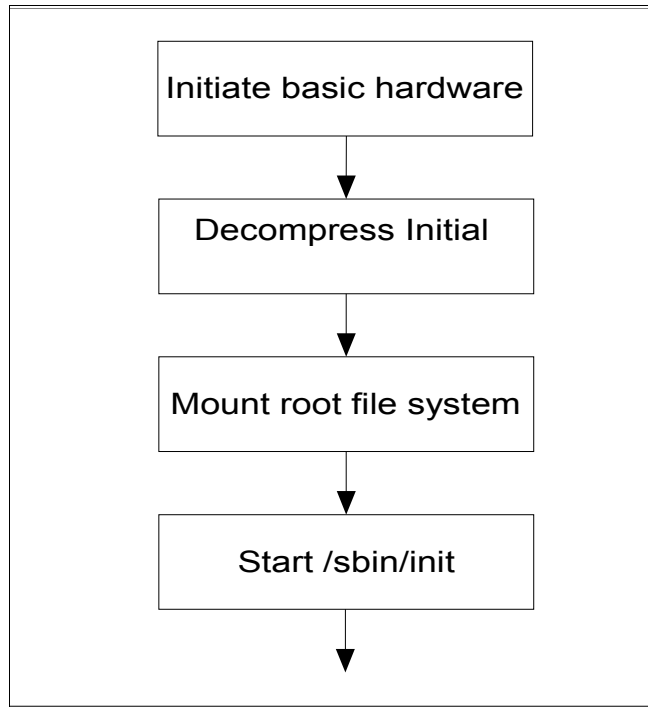


Figure 10-5 Kernel phase

Lets talk a little bit more about each kernel phase.

- Recognition and initialization of basic hardware



This step recognize and initiate basic hardware (processors, memory, console, etc) included in the kernel. Initialization of various kernel subsystem like SMP, Page table, NET is part of this step.

First of all, linux kernel decompress the kernel image on memory , then initialize several devices and subsystems by `start_kernel()` function.

Below, devices and subsystems initialized by kernel:

- Initialize kernel page table
- Outputs kernel banner
- Initialize physical memory
- Initialize CPU cacheL1/L2
- Initialize scheduler
- Registration of interrupt, trap and task gate
- Initialize RCU(Read-Copy Update)
- Initialize PID hash table
- Initialize CKRM (Class-based Kernel Resource Management)
- 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 all the time 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 some 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, linuxrc script is executed.

Processing of linuxrc script.

- Mounting proc, sys file systems
- Starting udev service
- Loading the driver modules /lib/modules or its sub directory
- Recognizing DASD devices and making them online
- Recognizing FCP channels and making FCP volumes online
- Recognizing LVM volumes and activate them
- Recognition of real root file system
  - Writing path info 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 init phase. At this point all file system are checked with 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 the mount is previously done as a root file system is saved in /initrd.

**Note:** In SLES9 and SLES10, the initrd is annulled because /initrd does not exist.

The reason to do the mount with read-only is to prevent the root file system 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 process from a script stored in the file `/etc/inittab`. This file usually has entries which 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 process

- ▶ Runlevel setting, remaining, start some daemons (following `/etc/inittab`)
- ▶ Run `/etc/init.d/boot.*` (Recognize LVM volumes, mount all file systems, etc.)
- ▶ Start daemons: `/etc/init.d/rc[0-6].d/*`

Let's detail a little bit about Init phase.

`/sbin/init` is executed when initialization by the kernel is completed, and init process is generated.

Initialization by init process.

#### **`/etc/inittab` configuration file.**

Various settings, initialization, and start processing are processed following `/etc/inittab`.

- ▶ Runlevel setting
  - Usually it 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`.
- ▶ Run the `/etc/init.d/boot`
  - Common system initialization that doesn't depend on runlevel
  - Time setting, Initializing swap device, Checking and mounting file systems, Initialize loop back device, Initialize quota, initialize `/proc` file system, etc.
- ▶ Run service scripts
  - Starting various demons according to runlevel: `/etc/init.d/rcX.d/*` scripts (X is 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 the starting of all daemons, `getty` is started.

Now, we will describe some parts of `/etc/inittab` configuration file.

`/etc/inittab` format

- `id:runlevels:action:process`

Main action in set item of `/etc/inittab`

- `initdefault` - Here is configured what is the default runlevel when the system starts.
- `bootwait` - the process will be executed during system boot..
- `once` -The process will be started once when the specified runlevel is entered.

The process that 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 `/etc/initab/` default at 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 /proc file system
- ▶ Mount /sys file system
- ▶ Mount /dev/pts
- ▶ Start blogd
- ▶ Execute various start scripts (these scripts are located at /etc/init.d/boot.d/ )
- ▶ Execute /etc/init.d/boot.local script

- ▶ Stop blogd

Executing /etc/init.d/boot scripts.

The scripts which placed on /etc/init.d/boot.d/ are called and executed by /etc/init.d/boot script. The following process (some) are executed by them:

- ▶ Load some kernel modules
- ▶ Activating swap file systems
- ▶ Activating SCPM(SUSE Configuration Profile Management)
- ▶ Update the library path
- ▶ Host name setting
- ▶ Activating loop back device and mounting it
- ▶ Time slice setting of scheduler
- ▶ IP setting (parameter settings related to N/W such as IP Forwarding)
- ▶ Kernel parameter setting specified in etc/sysctl.conf

**Note:** At /etc/init.d/boot.d, processes that start with Sxxx (e.g. S12SusSEfirewall2\_init ) are used to start when the machine boot, while processes that start with Kxxx (e.g. K20boot.rootfsck) are used to kill when the machine boot or is powered off

Starting Getty

When service start process have finished, getty program is started by init process, and the login prompt is displayed. As a result, login of the system becomes possible.

## 10.2 Where gather information regarding boot process.

There are many ways to collect information that are provide by the system during the boot. You can collect these informations accessing the log files /var/log/boot.msg, /var/log/ooboot.msg and /var/log/messages. Another way to get these informations is using the dmesg command.

When you must to use each one. See Figure 10-6

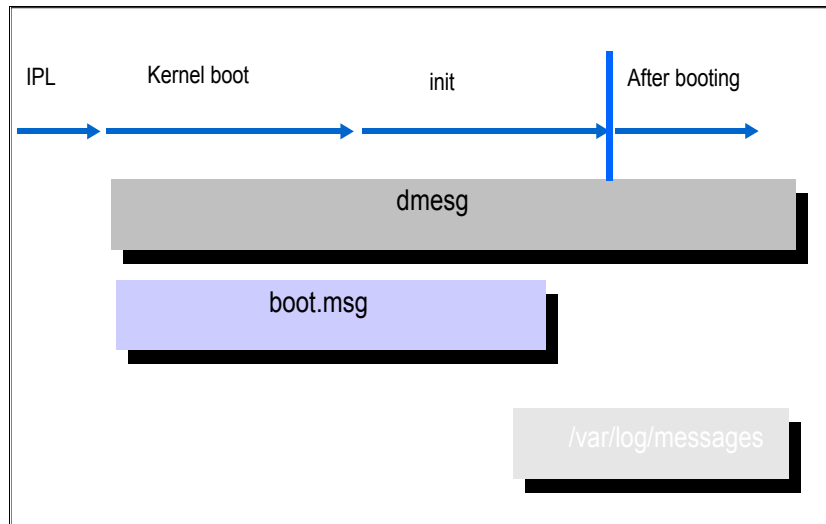


Figure 10-6 Gathering information.

The Booting problem distinction is done based on the boot message.

During the boot we can check the log output destination in 2 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 output destination:

- ▶ dmesg command
- ▶ /var/log/boot.msg file
- ▶ /var/log/boot.omsf file
- ▶ /var/log/messages

**Note:** During a problem determination, if the system doesn't respond can be used Message dump by Magic System Request Key (Magic SysRq key). Its send request to kernel directly and display message. The output is displayed in console and /var/log/message

Below more information about these options:

The `dmesg` command , means diagnostic message. The `dmesg` print out informations provided by kernel, but there is no information about IPL logs.

The file `/var/log/boot.msg` contain informations provided at the boot . At an initial phase, `/dev/console` é redirected for this file by the daemon `blogd`.

The file `/var/log/oboot.msg` contain informations about the last boot.

The file `/var/log/messages` contain informations that are printed out by the daemon `syslogd` .

At Example 10-3, see boot message to bootloader and kernel phases.

---

*Example 10-3 bootloader 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 understanding we split message boot logs according to the topics below.

- ▶ Starting Kernel (BootLoader phase)
 

Linux version 2.6.5-7.244-s390x (geeko@buildhost) (gcc version 3.3.3 (SuSE Linux))
- ▶ Display 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)

At Example 10-4, see boot message to 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 understanding we split message boot logs according 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 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/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 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)
kjournal 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
?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 understanding we split 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
```

► Runnig 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 filesystem

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 has completed.

Welcome to SUSE LINUX Enterprise Server 9 (s390x) - Kernel

2.6.5-7.244-s390x (tt

yS0).

itsolinux login:

## 10.3 Commands

Below are commands that can be used on boot process.

### **chkconfig**

updates and queries runlevel information for system services

the command `chkconfig` have 5 functions:

- add new services for management
- remove services from management
- list current startup information for services
- change the startup information for services
- checking the startup state of a particular service

### **dmesg**

print or control the kernel ring buffer

the command `dmesg` help users to print out their bootup messages.

### **mkinitrd**

creates initial ramdisk images for preloading modules

the command `mkinitrd` automatically loads filesystem modules, IDE modules, all `scsi_hostadapter` entries in `/etc/modprobe.conf`.

### **lsmod**

show the status of modules in the Linux Kernel

the command `lsmod` format the contents of the `/proc/modules`, showing what kernel modules are currently loaded.

### **insmod**

insert a module into the Linux Kernel

the command `insmod` insert a module into kernel.

### **rmmod**

remove a module from the Linux Kernel

the command `rmmod` remove a module from the kernel

### **modprobe**

add and remove 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**

bootloader tool for IBM S/390 and zSeries architectures.

Zipl can be used to prepare devices for initial program load (IPL) . It have the follow supported functions:

- booting a linux kernel with optional ramdisk and kernel command line
- asking 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 includes the following features:

- display a list of available configurations.
- allow to choose a configuration
- allow to specify additional kernel command line parameters.

### **fsck**

check and repair a Linux file system.

`fsck` is used to check and optionally repair one or more Linux file system. Normally, `fsck` command will try to handle filesystems on different physical disk drivers in parallel to reduce the total amount of time needed to check all of the filesystems.



## 10.4 Boot Problem examples

When we are talking about boot problem determination, there are basically 2 main problems

### **Linux can't finish booting**

A processing stops somewhere in the boot process. Many reasons can cause this problem. We can to say:

System stops in the single user mode, kernel panic, no response, etc..

### **The system does not work well, after boot is completed**

e.g login prompt is displayed.

The system have booted but Linux is not working properly. Some reasons can cause this problem. We can to say:

The service that should start automatically doesn't start, Some devices necessary for booting was not recognized, FCP devices cannot be recognized, LVM cannot be recognized or the network is not connected.

To a better understanding and definition where the problem is, We will use the follow diagram to help to identify which phase on the boot process have problems. Figure 10-7 on page 306

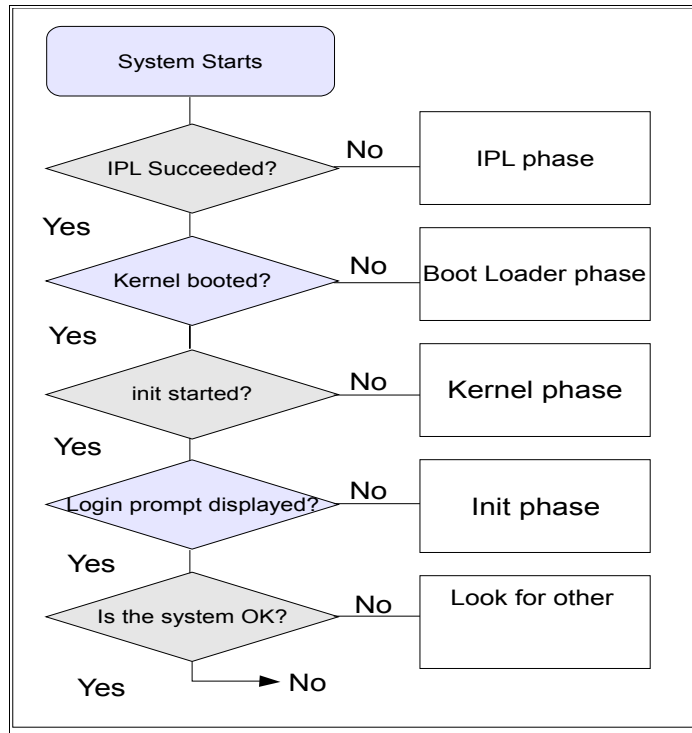


Figure 10-7 diagram to identify boot problem

At this point, We have 3 possibilities : This occurs immediately after install, after change the configuration or hardware failure.

### Immediately after installation

The installation might not be correctly completed due to the trouble of the installer. After the installation ends, Linux not boot it.

### After changing the configuration

When the LVM configuration change or some disk are added, etc, necessary processing (execution of zipl and re-creation of initrd etc) is not executed.

## 10.5 Scenarios

This topic describe some problems that can occur when you are booting your linux system. For all examples, lets use the same procedure to determine the solution.

First of all, we need to ask questions to presume what is causing the problem.After this let to investigate this point.

### 10.5.1 IPL failure

The following causes are thought when failing an IPL.

#### **Problem Description: IPL device is not found**

Questions to be answered:

- Is that correct IPL address?
- Is the IPL device address
- defined on the IOCP of that LPAR?(LPAR)
- attached the z/VM guest ?(Under VM)

#### **Problem Description: Boot loader code doesn't exist in the IPL device.**

Questions to be answered:

- Is that correct IPL address?
- Are zIPL configurations correct?
- Is the H/W trouble occurred in the specified IPL device? =>Do hardware problem determination

#### **Possible Solutions:**

- Confirm connection of the IPL device.
- Re-IPL with correct IPL Address.

Please execute the ziplt command from the rescue system, and introduce the boot loader code again when not solving it even if the above is executed.

*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 causes are thought when failing in the kernel booting

### **Problem Description: Kernel image in boot loader code is destroyed.**

Questions to be answered:

- Is the H/W trouble occurred in 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?
- Do you define “mem=” parameter in /etc/zipl.conf ?

### **Possible Solutions**

- Re-install boot loader code with correct kernel image file by using rescue system.
- Change 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 filesystems are not able to mount (excluding /)

Init cannot be skipped a pertinent process, and stops during processing when there is a problem involved DISK (ex. Partition problem)

**Problem Description: Failing in fsck.**

Questions to be answered:

- Is /etc/fstab description correct?
- Are the devices correctly recognized?
- Does not the file system break?

**Problem Description: Failing in mounting file system**

Question to be answered:

- Is /etc/fstab description correct?
- Are the devices correctly recognized?
- Does not the file system break?

**Possible Solutions**

- Modify entries of fstab
  - If there is a mis-setting, modify it
  - If a trouble in a file system occurs, comment out it temporarily or set fsck flag off (set it 0).
- Re-make initrd
  - The device including a necessary file system in booting makes it recognize with not coldplug but 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 is already mounted read-write.
```

Attention: Only CONTROL-D will reboot the system in this  
maintanancemode. shutdown or reboot will not work.

Give root password for login:

---

## 10.5.4 Failing in starting a service

The following two cases are thought when failing in the start of service.

### **Problem Description: Problem of automatic start setting**

Questions to be answered:

- Is the symbolic link file which name is started Sxx deployed in /etc/init.d/boot.d?
- Is the symbolic link linked with a correct start script?
- Is the symbolic link file which name is started S?? 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 doesn't start**

Questions to be answered:

- Has the required service started?
- Isn't there problem in the start script?

### **Possible Solution**

- Do automatic start setting of the service
  - It is set that service starts when booting by using the chkconfig command or the insserv command.

Remove the cause that service doesn't start referring to the boot message and syslog, etc.



## Case Study : Slow responding website

In this chapter, we will 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 for 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 on 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 that is being 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 Trade6 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 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:

- The 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.

► The 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.

► The DB2 database

The DB2 database contains relational tables regarding simulated customer accounts and stock transactions. We used DB2 LUW v8.2.

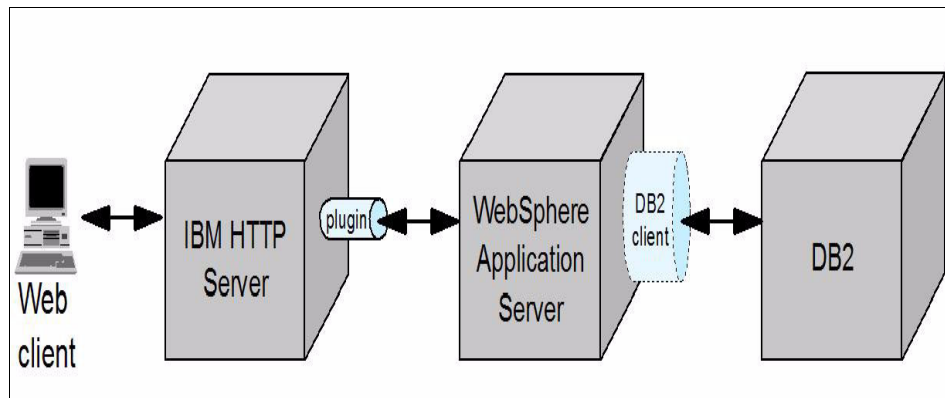


Figure 11-1 Components of a typical Trade 6 deployment

### 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 314). Our deployment choice is depicted in Figure 11-2 on page 314. 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

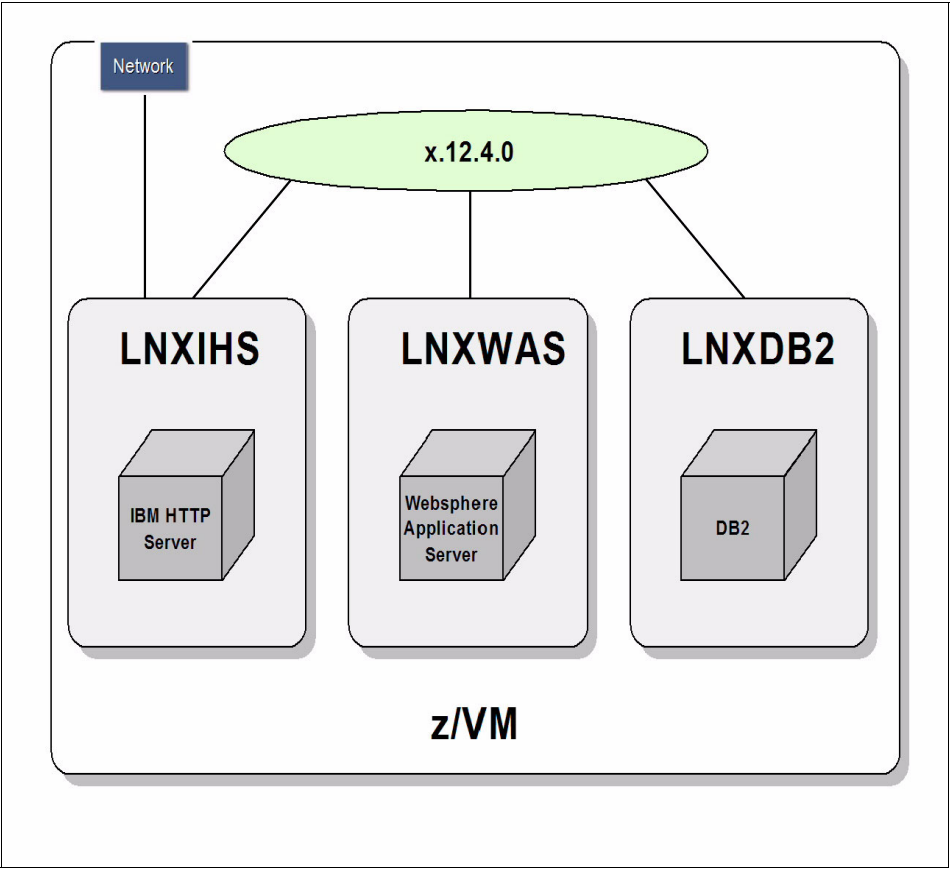


Figure 11-2 Three -tier deployment

In addition, when measuring utilization, the three-tier deployment option allows us to attribute specific resource usage to a specific task.

Table 11-1 Linux Servers and their configuration

| Server | Application           | processor | memory |
|--------|-----------------------|-----------|--------|
| LNXIHS | HTTP Server           | 1         | 512MB  |
| LNXWAS | WebSphere Appl Server | 1         | 3GB    |
| LNxDB2 | DB2 Server            | 1         | 2GB    |

### 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 on page 315.

*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 find out 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 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, 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 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 | <b>21.7</b> | <b>78.5</b> | 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 you will want to do is to collect information on the system. So we run the **top** command to see the dynamic resource utilization. In the 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 of course 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 | apldata     |
| 66   | root | 15 | 0  | 0     | 0    | 0   | S | 0.0  | 0.0  | 0:00.00 | pdflush     |

To get more detailed information on the swap, we ran the **vmstat** command to record the overall resource utilization of the system, which included memory, swap, io 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 on the "si" and "so" columns, run the command **man vmstat** . Notice also that the CPU is almost idle (on average, 97%) most of the time.

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 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 3GB. Then we started the workload with the same number of users we opted for as in the previous workload runs.

When the load reached the peak, we wanted to verify whether the problem was re-occurring. Also we are monitoring the workload simulator for any of the errors that were generated earlier.

Example 11-7 System utilization with 3GB 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 |

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.

11.2.3 What if the load is increased - does the problem re-appear?

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 on the servers.

After some time, when the load was at its peak, there were once again some “timed-out” messages displayed on the Workload Simulator. But unlike the previous workload simulation, this time the `vmstat` tool didnt report any swapping or high memory usage (see Example 11-8).

Example 11-8 vmstat report without any swapping with 3GB 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 |

---

And from Example 11-8, it appears that the system is stable and 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 lots of 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 simulators 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 hierachy of paging devices. Expanded storage is the high speed paging device and DASD the slower one where block paging is done.

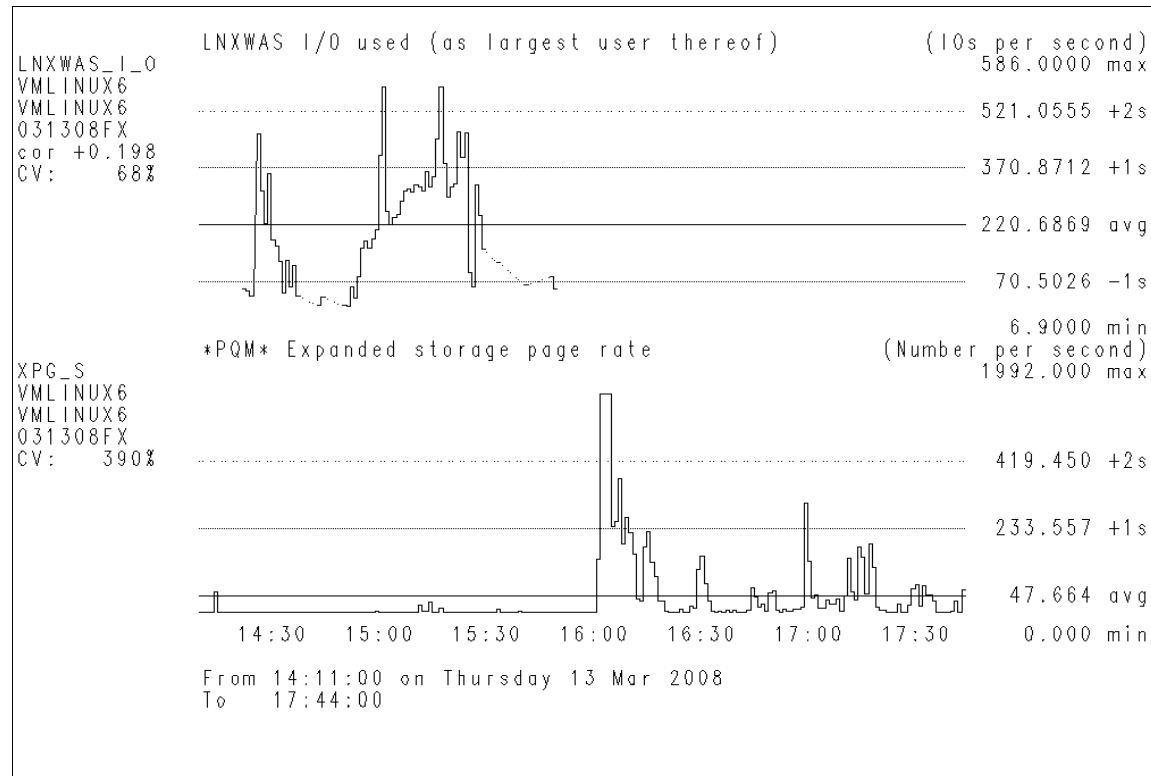


Figure 11-3 Example Expanded Stor Page rate compared with I/O operation

This means 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 that there is heavy paging to expanded storage. However, expanded storage is sufficient 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.

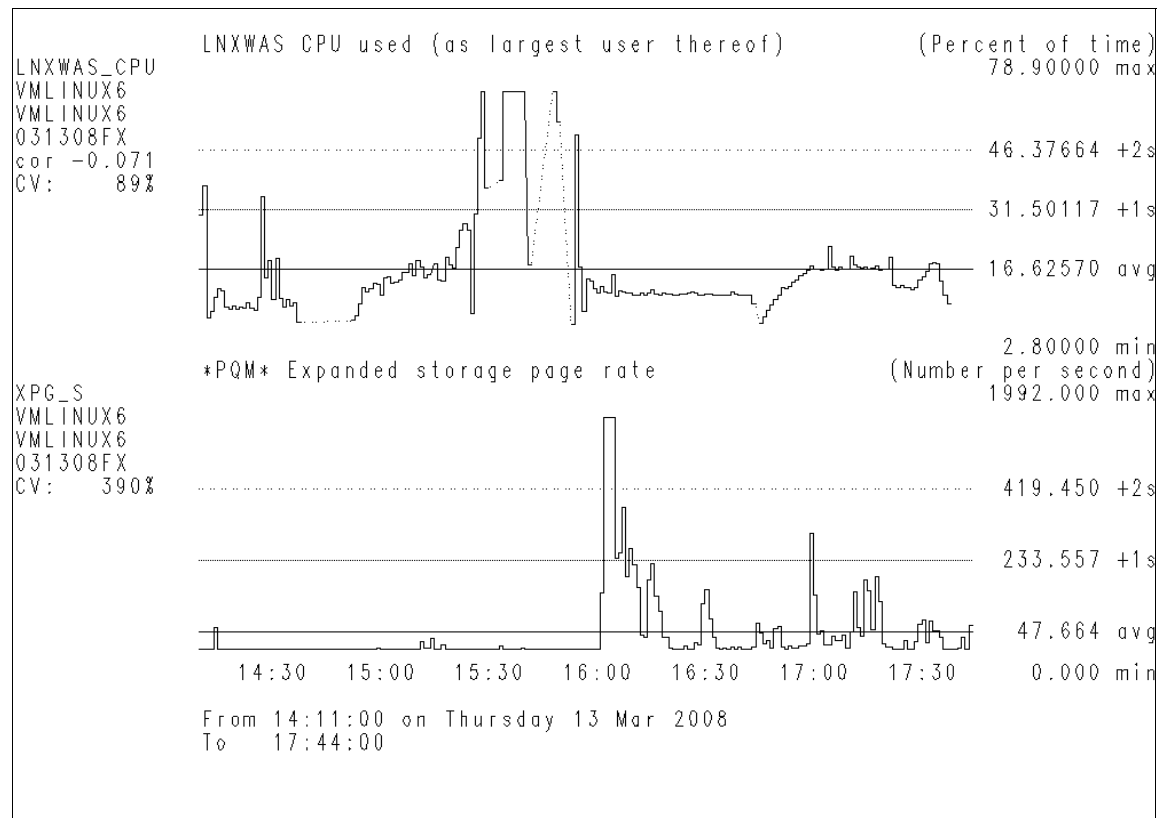


Figure 11-4 Example Expanded Storage page rate compared with CPU used

The Figure 11-4, shows an 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.

The next obvious statistical data of our interest would be the System Page rate, as we can see in Figure 11-5, we can say that there is heavy system paging happening during the narrowed down timings.

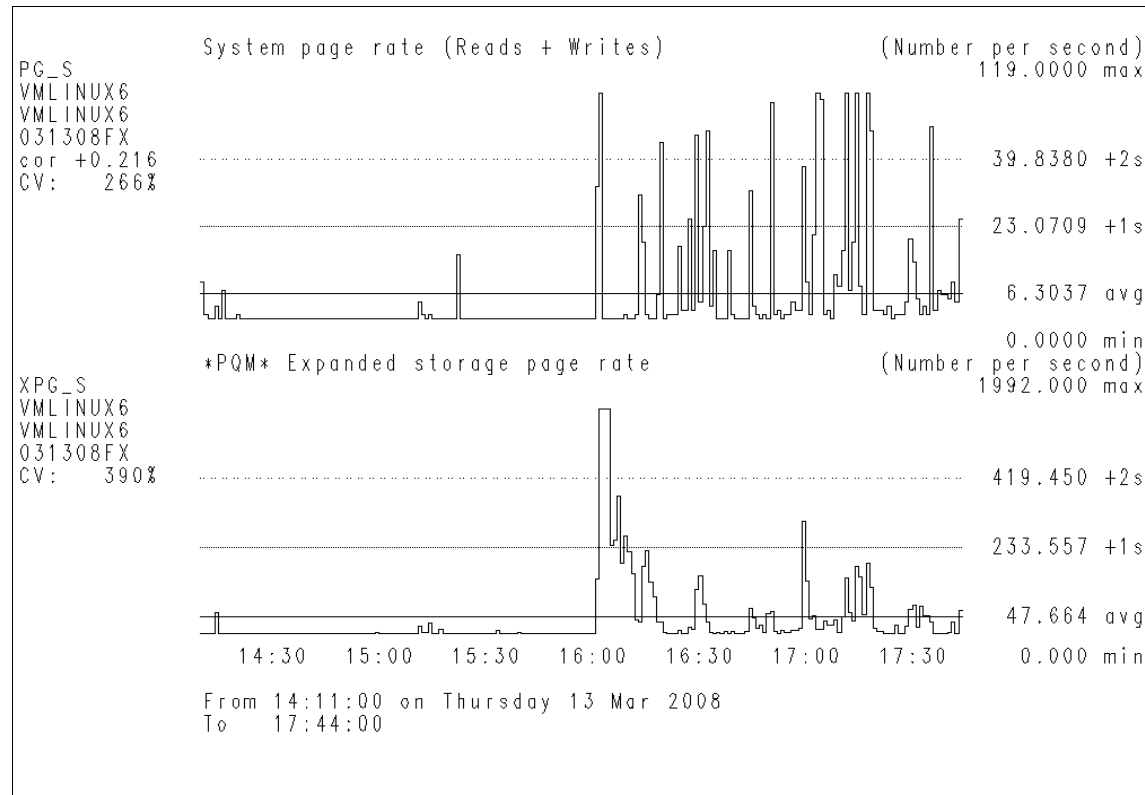


Figure 11-5 Example Expanded Stor page rate with System Page rate

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 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 can't fit everything in real storage at once. Therefore paging increases are caused by

- Decrease in available storage
- Increase in storage requirements
- Or combination of the previous points

QUERY FRAMES is the recommended way of checking storage, but it's only online. In our case, since the Performance toolkit provides the same information with 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.

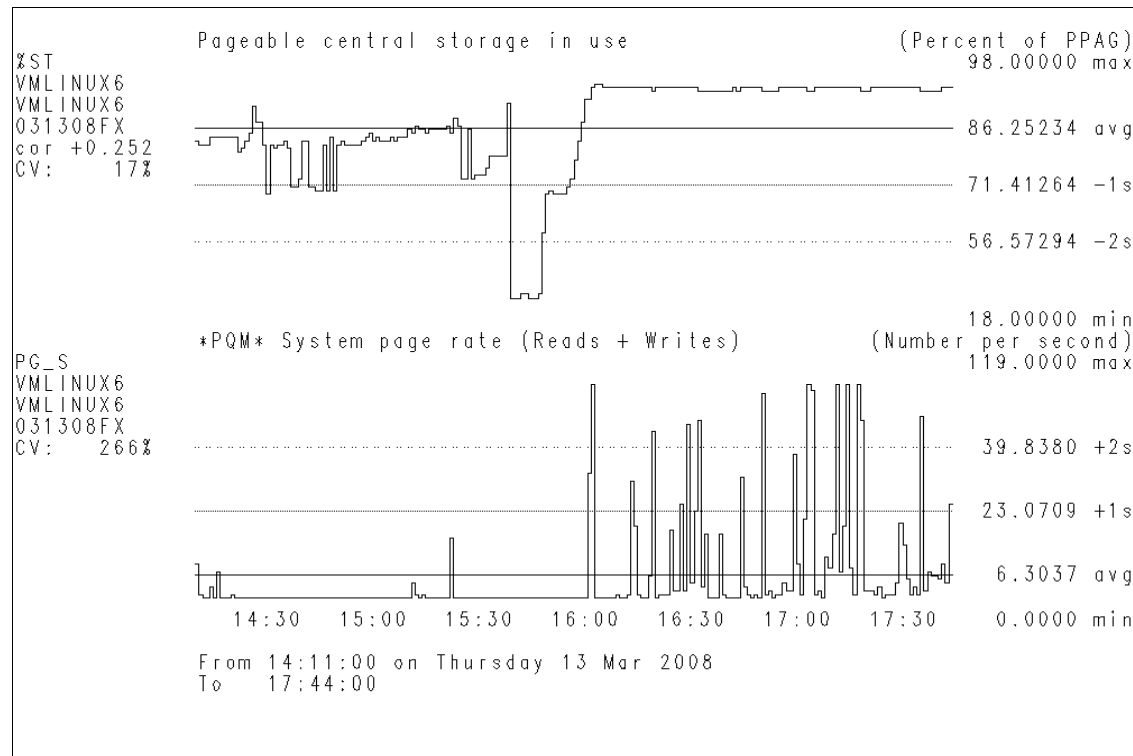


Figure 11-6 Example System Page rate compared with Pageable Central Stor in use

In the Figure 11-6, we compared with Pageable central storage in use. As we can see, the Pageable Central 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 central storage gets filled.

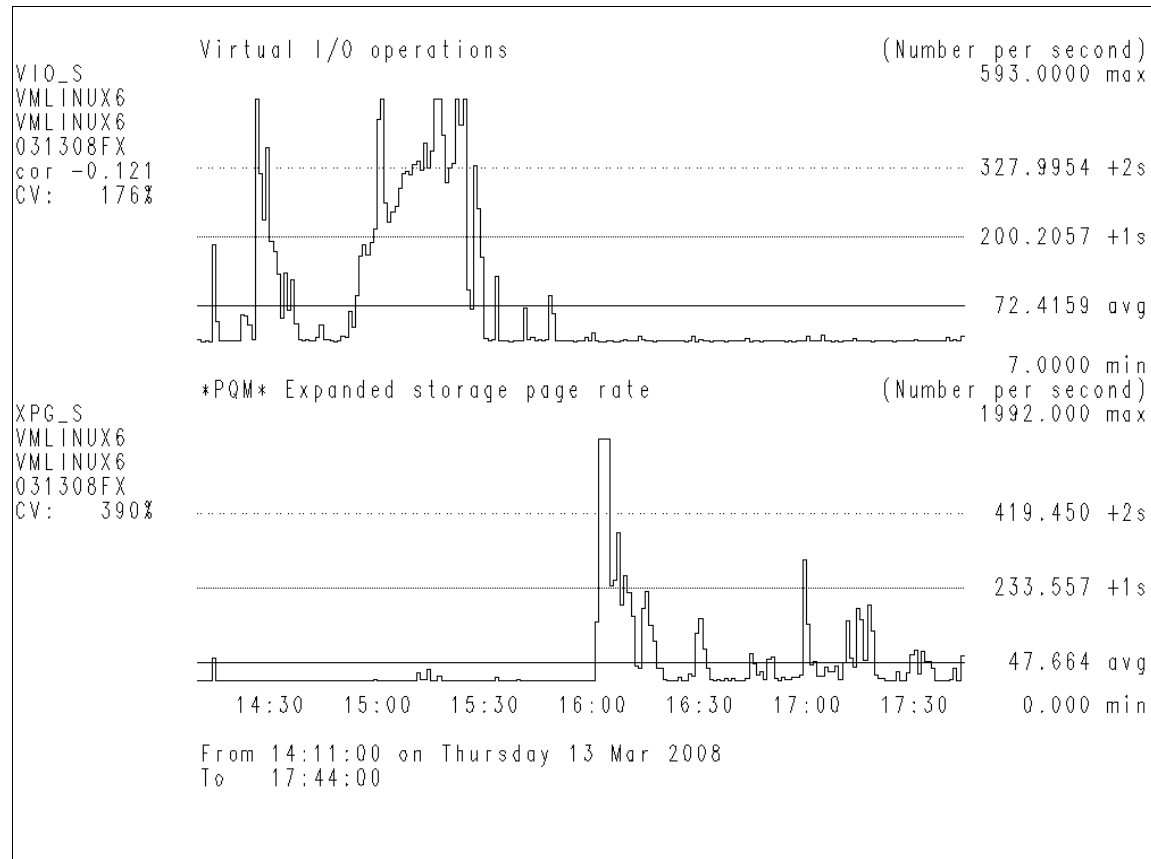


Figure 11-7 Example Expanded Page rate compared with I/O operations

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 not I/O operation, which means that the Processor is spending most of its time paging. This sharp variations in the resource utilization indicates some erratic and serious system behavior.

## 11.4 Emergency Scanning

Further when we drill down the dynamic data provided by 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 a 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 of up 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 has to resort to emergency scanning to find the storage pages that it requires to complete the I/O task.

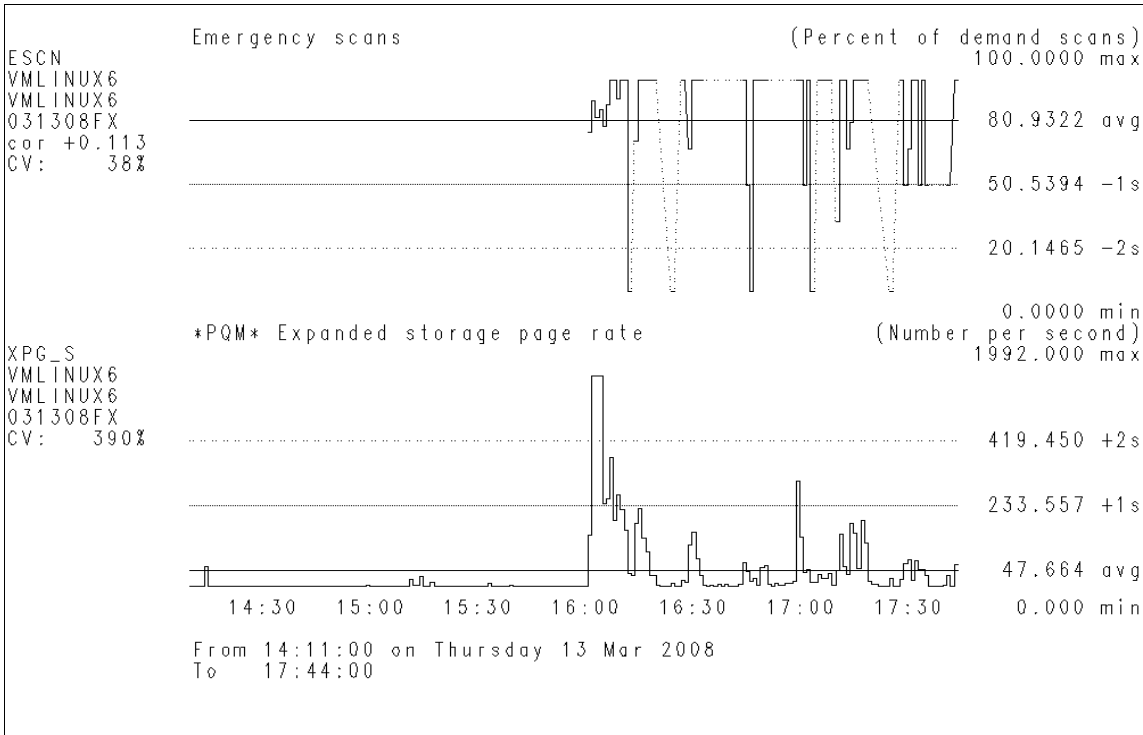


Figure 11-8 Example Emergency scanning compared with Exp Storage Page rate

Figure 11-8, shows a where there are heavy emergency scanning happening at the same time period when we noted that there are some Expanded Storage Paging.



Figure 11-9, shows an increase in the Page rate and at the same time, we can see that z/VM has entered in to an emergency scanning mode for release pages.

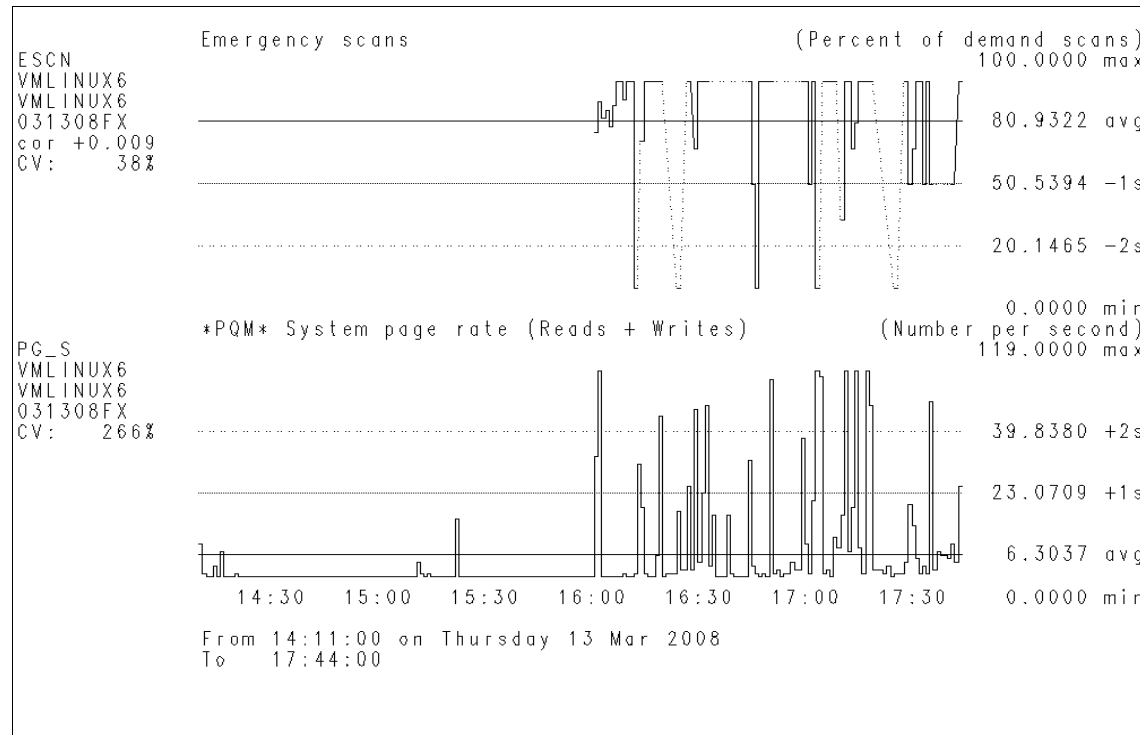


Figure 11-9 Example Emergency Scanning compared with System Page rate

**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 (Applicatoion 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.

## IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 332. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *????full title???????, xxxx-xxxx*
- ▶ *????full title???????, SG24-xxxx*
- ▶ *????full title???????, REDP-xxxx*
- ▶ *????full title???????, TIPS-xxxx*

## Other publications

These publications are also relevant as further information sources:

- ▶ *????full title???????, xxxx-xxxx*
- ▶ *????full title???????, xxxx-xxxx*
- ▶ *????full title???????, xxxx-xxxx*

## Online resources

These Web sites are also relevant as further information sources:

- ▶ Description1  
<http://?????????.???./???/>
- ▶ Description2  
<http://?????????.???./???/>
- ▶ Description3  
<http://?????????.???./???/>

## 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](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)

# Index

## Symbols

!!XGEN\_DONT\_EDIT\_THIS! 107–118, 120,  
122–125, 129–130, 132–134, 138, 140, 143, 147,  
149, 152–156

## A

Application layer 89

## B

barrier-based sync 300  
blks 265, 297  
boot process 264  
boot.mult ipath 209  
bottom line 39

## C

cached 58  
Case study 260  
central storage (CS) 325  
Channel Subsystem 69  
command privilege (CP) 228  
Common I/O  
    Layer 239  
complex problem 17  
configuration file 268  
    per-host basis 268  
Consensus term 15  
control program (CP) 22  
Core Dump  
    classic example 85  
Core dump 84  
CP command  
    trace 28  
CPU  
    basis 172  
    time  
        172  
    type 23  
    utilization 51, 61, 163  
cpu time 229

## D

DASD  
    device 78, 203, 290  
    getting I/O statistical information 69  
    statistics option 67  
dasd driver 187  
DASD dump 204  
DASD Statistic 187  
DASD volume 41, 81  
    ordered list 41  
dasdview command 204  
DB2 database 214, 313  
DB2 Server 314  
Dec 18 19  
    19  
        04 xxxxxxxx kernel 254  
default ERP 246  
device 0.0.3600 260  
Device D437 47  
    Detailed Analysis 47  
device driver 101, 239  
device file 239–240  
Diag. X'98 45, 215  
disk B 33  
disk I/O  
    bottleneck 183  
    operation 183  
disk storage (DS) 179  
dispatch list 26, 228  
dmesg command 294  
dmesg print 296

## E

elig 43, 214  
Eligible List  
    class users 229  
eligible list 25, 228  
EME 44, 215  
end user 158, 233  
Environment setup 85  
error message 5, 252, 320  
error recovery 190, 243  
    action 191

ESA 44  
 etc/init.d/boot script 293  
 EXPAN-001 E2-00000 24, 230  
 EXPAN-001 E3-00000 24, 230  
 Expanded storage  
     comparative chart 323  
 expanded storage (ES) 23, 176, 321  
 ext3 filesystem 172, 298  
 EXT3 FS 300  
 Extended IPv6 support 277  
 Extended Link Service (ELS) 255

## F

FCP connection 184, 255  
 FCP device  
     recognition 258  
     status 250  
 FCP LUN 73  
     I/O statistics 78  
 FCP port 258  
 file system 8, 288, 290  
 filesystem cache 179  
 filesystems 304  
 Firewall rule 277  
 follow-up action 1, 18  
 FORMAT F 254  
 fsck command 290

## G

granular information 21, 57  
 Graphical Data Display Manager (GDDM) 38

## H

H/W error 253–254  
 H/W problem 256  
 H/W trouble 252, 307  
 HCPGIR450W CP 34, 308  
 Host Bus Adapter (HBA) 189  
 HTTP server 312

## I

I/O act 45, 215  
 I/O activity 64, 189  
 I/O Device 238  
 I/O device  
     load 41, 186  
     performance 37

I/O interruption 244  
 I/O load 37  
 I/O operation 164, 239, 322  
 I/O rate 43, 214  
 I/O request 244  
 I/O time 68, 188  
 IBM Tivoli OMEGAMON XE 50  
 IBM z/VM  
     Performance Toolkit 21, 40, 321  
 IBM z/VM Performance Toolkit  
     feature 37  
     initial screen 40  
     main menu 48  
     main screen 41  
     menu 43  
 IBM z/VM Performance Toolkit commands  
     redispatch 48  
 Ideal size 220  
 Init phase 284, 290  
     little bit 291  
 Initial Program Loader (IPL) 284  
 initial set 42  
 Input/output (I/O) 59, 173  
 Inter process communication (IPC) 65  
 IP address 278  
 IP Forwarding 294  
 IPL address 307  
 IPL device 285  
 IWL0046W HTTP 316

## J

JVM side 177

## K

kernel panic 33, 78, 305  
     system crashes 78  
 Kernel phase 288  
 kernel space  
     callgraph feature 72  
 kernel version  
     2.6.16.21 74  
     2.6.5 74  
 ksoftirqd 172

## L

Linux 3, 21, 52, 57, 157, 201, 237, 268, 284, 311  
 Linux commands

- dasdview 204
- lscss 202
- lsdasd 203
- zipl 203
- Linux distribution 74, 268
- Linux environment 57
- Linux guest 32, 232, 313, 328
  - example SRM LDUBUF setup 233
  - paging volume 234
  - virtual machine size 313
- Linux image 58, 312
- Linux kernel
  - thread 172
- linux kernel 277, 286, 319
- Linux side 178
- Linux system 30, 33, 64, 70, 164, 202, 281, 307
  - increases 172
  - move 164
  - running code 70
- Linux tools
  - callgraph 72
  - iostat 65
  - ipcs 65
  - Netstat 66
  - Opcontrol 71
  - OPProfile 70
  - ps 64
  - sysstat 60
  - top 64
  - tunedasd 69
  - vmstat 59
- linuxrc script 203, 290
- log level 99, 242
  - zFCP driver 104
- Logical Volume
  - Manager 184
- ls command 93
- lscss command 202, 248
- lsdasd command 203, 250
- lszfcp command 207
- ltrace utility 95
- LVM command 17
- LVM volume 290
  - group 299

## M

- Machine type 257
- main stor 220

- Ideal size 220
- Max. size 220
- major/minor number 243
- Mar 10 12
  - 54
    - 05 linux kernel 277
  - 56
    - 32 linux SuSEfirewall2 277
  - 57
    - 40 linux SuSEfirewall2 278
- Mar 10 13
  - 00
    - 57 lxinst ifup 278
- MDISK cache 220
- Memory usage 50, 64, 224
- memory utilization 64, 163
- middleware 158
- min 64, 316
- MTU size 193

## N

- named saved segments (NSS) 304
- Netfilter message 277
- NETIUCV driver 277
- network interface 66, 260
  - displays statistics 67
- networking option 163, 193
- ni 59, 321

## O

- OMEGAMON display 50
- operating system 158, 284, 317, 319
- Organizing the system (OS) 4, 89
- OSA Address Table (OAT) 138
- OSA card 194
- OSA device 242

## P

- paging CHPIDs 234
  - one-to-one relationship 234
- paging operation 166
- Performance problem
  - CPU constraints 161
  - I/O resource constraint 161
  - memory 161
  - memory constraint 161
  - network utilization 163

- operating system level 158
- resource bottlenecks 158
- software problems 158
- performance problem 28, 55, 157–158
- Performance Problem Determination 157–158
  - General flow 157
- Performance Toolkit 8, 21, 168, 321, 325
  - paging activity 325
- PF-key usage 39
- physical volume 264, 298–299
- problem determination 1, 8, 21, 57, 157–158, 227, 267, 283, 311–312
  - analysis time 93
  - basic steps 9
  - first step 11
  - IBM z/VM Performance Toolkit 49
  - important aspect 14
  - meaningful data 42
- proc directory 101
- proc file system 291

## Q

- qdio mode 240
- query srm 27
- Queue Statistic 43, 214
- Queued Direct I/O (QDIO) 239

## R

- Read-Copy Update (RCU) 289
- reader file 31, 34
- real storage 231, 317, 322
- Redbooks Web site 332
  - Contact us 3
- Reserved pgs 45, 215
- resource utilization 55, 58, 159, 317
  - historic data 164
- response time 37, 59, 183

## S

- S/W version 259
- same number 319
- Sample output 182
- sar 60–61
- SCSI device 206, 243, 299
  - composition information 248
- SCSI disk
  - dump tool 81

- recognition 258
- SCSI trace 103
  - log level 105
- Secure Access Key (SAK) 97
- segmentation fault 90
- Service Level Agreement (SLA) 4
- Service time 47, 173, 186
- SET QUICKDSP 229
- Setup Problem 267
- single point of failure (SPOF) 2
- SLES9 SP3 207, 261
- software/program 285
- specified runlevel 291
- SPOF failure 3
- SPOOL pg 45, 215
- SRM STORBUF
  - 300 232
  - setting 234
  - value 232
- st 49, 60, 223, 239, 299, 316
- st 0 60
- st Mem 64, 316
- stand-alone dump 5, 80
- state matching 277
- storage page 42, 222, 327
- Storage problem 201
- storage server 186
- Storage Subsystem 69, 183, 185
- storage subsystem
  - practical limit 186
- SUSE Linux Enterprise Server
  - 10 Gallium 74
  - 9 SP3 74
- sy 75, 316
- symbolic link 258, 310
- system call 78, 80
  - statistic 96
  - status 89
- system call status 80
- System z 3, 20, 27, 57, 157, 164, 201, 237–238, 268, 284, 312
  - Linux installation 268

## T

- t 24, 85, 173, 204, 293
- T=0.33/0.73 15
  - 24
    - 45 36



## TEACHOR1 QDIO

0601 OSA 45

0650 OSA 45

three-tier deployment 313

trace information 102

trans. time 43, 216

turn around time (TAT) 2

**U**

user LNXDB2 215

Detailed data 215

User resource usage

page 43

User Status 43, 214

userid LXOR4 44

detailed status 46

Utilities Reference 26

**V**

var/log directory 275

var/log/boot.msg file 295

VF emulation 45, 215

VF overhead 45, 215

VG 212

virtual machine

CP class 23

performance monitors 39

virtual machine (VM) 23, 157, 228–229, 313

VM guest 163, 307, 319

VM user 308

vmstat output 168–169, 318

volume table of contents (VTOC) 204

vswitch 22

**W**

Working Set Size

Large values 179

Working Set Size (WSS) 179

Workload Simulator 315

**X**

XAL/S (XS) 49, 223, 327

XSTORE page 215

**Z**

z/VM system 23, 52, 136, 224, 321

overall status 23

real time performance analysis 38

z/VM system tools

cp close console 31

cp query 30

dumpload 34

indicate 23

indicate load 23

indicate paging 24

indicate paging wait 24

indicate queues 25

paging wait 24

q trace 29

query alloc page 27

query srm 27

trace 29

vmdump 32

zFCP

steps to gather statistical information 74

zFCP Statistic 73–74, 189

zFCP trace 103

log level 104



To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5" spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(--->Hide:)>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine.frm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.

**7599spine.frm    339**



# Problem Determination for Linux on System z

(1.5" spine)  
1.5"<=> 1.998"  
789 <=> 1051 pages



# Problem Determination for Linux on System z

(1.0" spine)  
0.875"<=> 1.498"  
460 <=> 788 pages



# Problem Determination for Linux on System z

(0.5" spine)  
0.475"<=> 0.875"  
250 <=> 459 pages



# Problem Determination for Linux on System z

(0.2" spine)  
0.17"<=> 0.473"  
90 <=> 249 pages

(0.1" spine)  
0.1"<=> 0.169"  
53 <=> 89 pages

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5" spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine.frm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.

**7599spine.frm 340**



# Problem Determination for Linux on System z

(2.5" spine)  
2.5"<->nnn.n"  
1315<-> nnnn pages



# Problem Determination for Linux on System z

(2.0" spine)  
2.0"<->2.498"  
1052 <-> 1314 pages





Draft Document for Review June 13, 2008 9:57 pm

# 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**

**Case study using  
what you learn here**

This IBM Redbooks publication provides a problem determination methodology and the tools to help the reader in an easy to read self-help manual. The intention of this IBM Redbooks publication 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.

This book also lists and describes some of the tools that are available to assist in diagnostics and discuss under what circumstances you would use each tool.

## **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](http://ibm.com/redbooks)