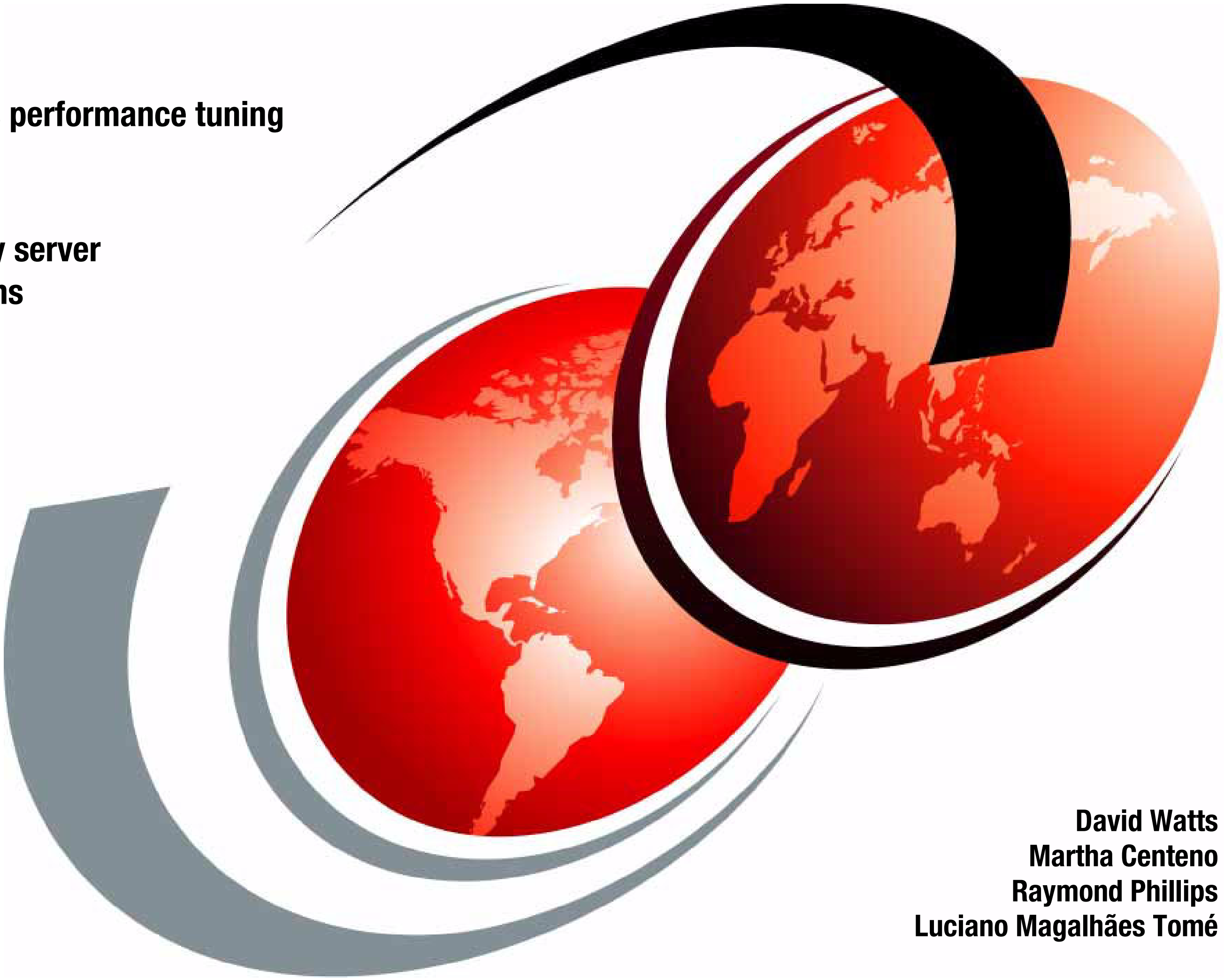
Tuning SUSE LINUX Enterprise Server on IBM *e*server xSeries Servers

Describes ways to tune the operating system

Introduces performance tuning tools

Covers key server applications



David Watts
Martha Centeno
Raymond Phillips
Luciano Magalhães Tomé

Redpaper

ibm.com/redbooks

International Technical Support Organization

Tuning SUSE LINUX Enterprise Server on IBM eserver xSeries Servers

July 2004

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

First Edition (July 2004)

This edition applies to SUSE LINUX Enterprise Server 8 and 9 running on IBM @server xSeries servers.

Contents

Notices

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates 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. 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 IBM's application programming interfaces.

Trademarks

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

AIX®
BladeCenter™
DB2®
Domino®
@server®
@server®
IBM®
Lotus®
Lotus Notes®
Notes®
OS/2 WARP®
OS/2®
Redbooks™
Redbooks (logo) ™
ServeRAID™
xSeries®

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

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

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

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

SUSE LINUX is a registered trademark of SUSE LINUX AG, a Novell, Inc. company and the information and screen captures were used with permission from Novell, Inc.

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

Preface

Linux is an open source operating system developed by people all over the world. The source code is freely available and can be used under the GNU General Public License. The operating system is made available to users in the form of distributions from companies such as SUSE LINUX. Whereas some desktop Linux distributions can be downloaded at no charge from the Web, the server versions typically must be purchased.

IBM® has embraced Linux, and it is now recognized as an operating system suitable for enterprise-level applications running on IBM @server® xSeries® servers. Most enterprise applications are now available on Linux as well as Microsoft® Windows®, including file and print servers, database servers, Web servers, and collaboration and mail servers.

With the use in an enterprise-class server comes the need to monitor performance and, when necessary, tune the server to remove bottlenecks that affect users. This IBM Redpaper describes the methods you can use to tune SUSE LINUX Enterprise Server, tools that you can use to monitor and analyze server performance, and key tuning parameters for specific server applications.

The team that wrote this Redpaper

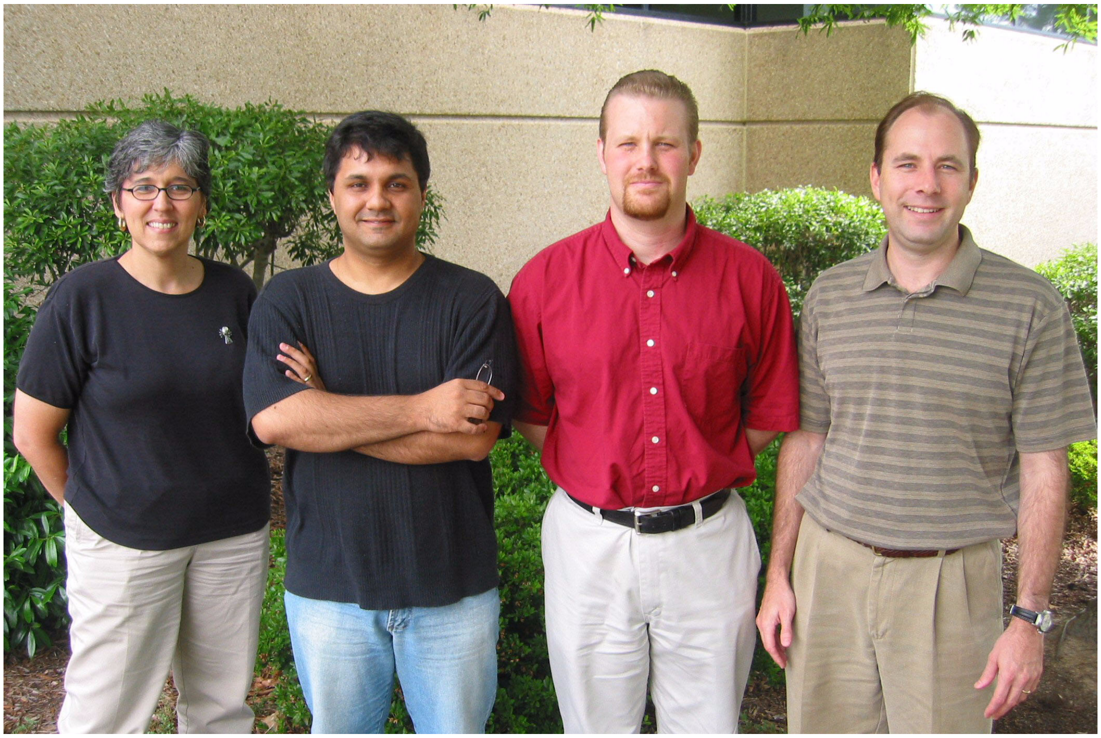
This Redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

David Watts is a Consulting IT Specialist at the International Technical Support Organization in Raleigh. He manages residencies and produces IBM Redbooks™ on hardware and software topics related to xSeries servers and associated client platforms. He has authored more than 25 Redbooks and Redpapers; his most recent books include *IBM @server xSeries 445 Planning and Installation Guide*. He holds a Bachelor of Engineering degree from the University of Queensland (Australia) and has worked for IBM for more than 14 years. He is an IBM @server Certified Specialist for xSeries and an IBM Certified IT Specialist.

Martha Centeno is an Advisory Software Engineer in the IBM @server xSeries Performance lab in the Research Triangle Park in North Carolina. Over the past 10 years in the performance group, she has been involved in the publication of leading benchmark results of xSeries servers in the SAP and TPC-W environments. Her current focus is on the performance of Oracle RAC on xSeries servers and BladeCenter™ in Linux environments.

Raymond Phillips is an IBM I/T Architect in IBM Global Services USA. He has more than 10 years of experience in Information Technology, mainly in the financial sector, with the last five working on the Washington Mutual Account. His areas of expertise include building enterprise solutions utilizing Linux and Windows.

Luciano Magalhães Tomé is a Systems Developer working for EDS in Brazil. He has seven years of experience in UNIX Systems Administration and development in C/C++. Currently, he provides support to other EDS professionals in the USA, UK, and Italy who use the IT automation tool *Opsware*.



The team (l-r): Martha, Luciano, Raymond, David

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

IBM: Jeff Brenner, Pat Byers, Jere Cline, Rufus Credle, Art Fisher, Amy Freeman, Cheryl Gera, Wendy Hung, Grant Hutchison, Bruce Jones, Darryl Miles, Kiron Rakkar, Daryl Stokes, John Tran, William Tworek, Peter Wong

SUSE LINUX: Dennis Conrad, Michael Hager, Susanne Oberhauser

Novell: Sue Goodwill

EDS Brazil: Eduardo Kobayashi, Alexandre Augusto, Eduardo Wong

Vesper Brazil: Fabio Capelari, Nelson Speed Santos

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HZ8 Building 662
  P.O. Box 12195
  Research Triangle Park, NC 27709-2195

1

Tuning the operating system

By its nature and heritage, the Linux distributions and the Linux kernel offer a variety of parameters and settings to let the Linux administrator tweak the system to maximize performance.

This chapter describes the steps you can take to tune SUSE LINUX Enterprise Server[1]. The objective is to describe the parameters that give you the most improvement in performance. It also gives you some basic understanding of the techniques that are used in Linux, including:

- Linux memory management
- Page partitions in Linux
- File systems and their effect on performance
- Disabling unnecessary daemons
- Tuning parameters using sysctl

This chapter has the following sections:

Restriction: Recompiling the kernel or the addition of kernel modules other than those provided by SUSE voids standard SUSE service and maintenance liabilities. You should contact SUSE LINUX to request an individual service agreement on such a non-standard setup.

[1] SUSE LINUX is a registered trademark of SUSE LINUX AG, a Novell, Inc. company, and the information and screen captures were used with permission from Novell, Inc.

1.1 Disabling daemons

There are daemons (background services) running on every server that are probably not needed. Disabling these daemons frees memory, decreases startup time, and decreases the number of processes that the CPU has to handle. A side benefit to this is increased security of the server because fewer daemons mean fewer exploitable processes.

By default, several daemons are started that can be stopped safely on most servers.

By default, SUSE LINUX Enterprise Server 8 has the following daemons listed in Table 1-1 started, and you should consider disabling these within your environment if possible. For a more detailed explanation of these daemon, refer to the YaST tool as shown in Figure 1-2 on page 3.

Table 1-1 Tunable daemons started on a default install

Daemon	Description
alsasound	Sound daemon
isdn	ISDN modem support
hwscan	Detects and configures hardware changes
portmap	Dynamic port assignment for RPC services (such as NIS and NFS)
postfix	Mail Transport Agent
splash	Splash screen setup
fbset	Framebuffer setup
splash_late	Starts before shutdown
splash_early	Kills animation after the network starts
xdm	X Display manager

You can temporarily stop most daemons with the *stop* parameter. For example, to stop the sendmail daemon immediately, enter the following command as root:

```
/etc/init.d/sendmail stop
```

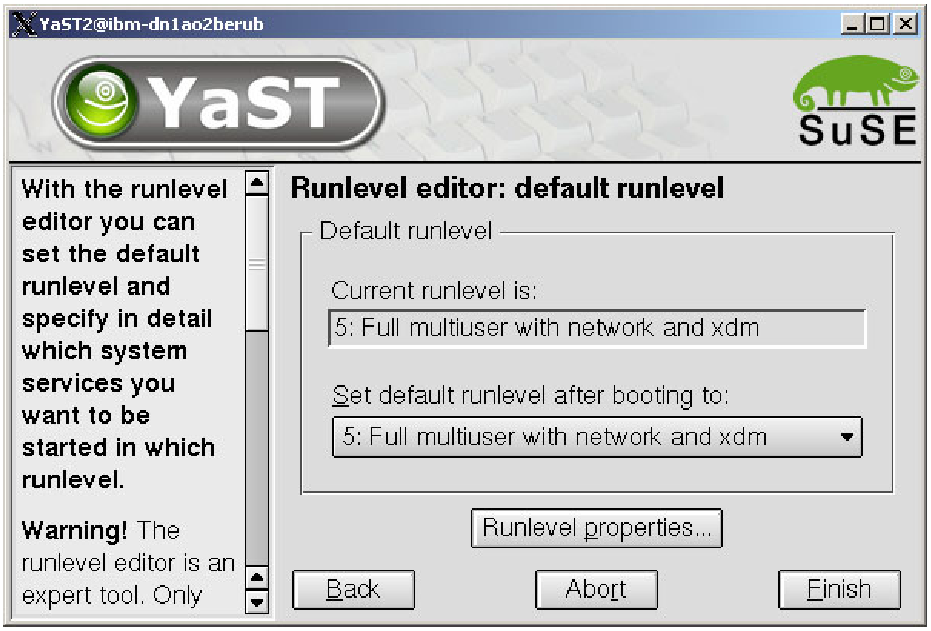
In addition, SUSE LINUX Enterprise Server 8 has three ways to work with daemons:

- A text-based UI: **`/sbin/yast runlevel`**
- A graphical user interface (GUI), YaST2, which can be started either with the command **`/sbin/yast2 runlevel`** or by clicking **Browse: YaST/** → **YaST modules** → **System** → **Runlevel editor**, as shown in Figure 1-1 on page 3
- The **`/sbin/chkconfig`** command

If you do not want the daemon to start the next time the machine boots, enter the following command as root:

```
/sbin/chkconfig -s sendmail off
```

The YaST2 GUI also lets you perform this action.



With the runlevel editor you can set the default runlevel and specify in detail which system services you want to be started in which runlevel.

Warning! The runlevel editor is an expert tool. Only

Runlevel editor: default runlevel

Default runlevel

Current runlevel is: 5: Full multiuser with network and xdm

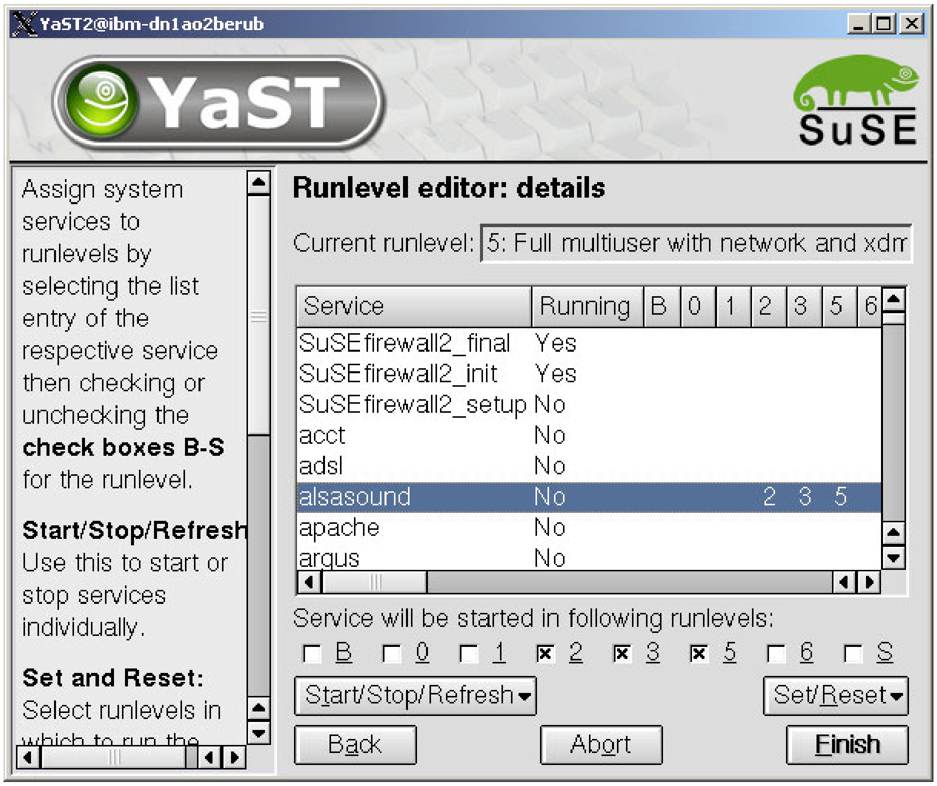
Set default runlevel after booting to: 5: Full multiuser with network and xdm

Runlevel properties...

Figure 1-1 YaST Runlevel editor

To change the startup state of a daemon, do the following:

1. From the main window, click **Runlevel properties**, which displays the Runlevel editor details window. See Figure 1-2.



Assign system services to runlevels by selecting the list entry of the respective service then checking or unchecking the **check boxes B-S** for the runlevel.

Start/Stop/Refresh Use this to start or stop services individually.

Set and Reset: Select runlevels in which to run the

Runlevel editor: details

Current runlevel: 5: Full multiuser with network and xdm

Service	Running	B	0	1	2	3	5	6
SuSEfirewall2_final	Yes							
SuSEfirewall2_init	Yes							
SuSEfirewall2_setup	No							
acct	No							
adsl	No							
alsasound	No				2	3	5	
apache	No							
argus	No							

Service will be started in following runlevels:

□ B □ 0 □ 1 ☒ 2 ☒ 3 ☒ 5 □ 6 □ S

Figure 1-2 YaST Runlevel editor properties

2. From within this window, you can change the start state of a daemon by checking or clearing the appropriate runlevel box.
3. Click **Finish** to save the runlevel states.

Tip: You must click the **Start/Stop/Reset** button to stop a service immediately, as changing the runlevels will not take effect until the next startup.

1.2 Shutting down the GUI

Whenever possible, do not run the graphical user interface on a Linux server. Normally, there is no need for a GUI on a Linux server. All administration tasks can be achieved by the command line, redirecting the X display or through a Web browser interface. There are several different useful Web-based tools (for example, webmin, Linuxconf, and SWAT).

If there is really a need for a GUI, start and stop it as needed rather than running it all the time. In most cases the server should be running at runlevel 3, which does not start the GUI when the machine boots. If you want to restart the Xserver, use **`startx`** from a command prompt.

1. Determine which run level the machine is running with the command:

   ```
   runlevel
   ```

   This prints the previous and current run level (for example, `N 5` means that there was no previous run level (N) and that the current run level is 5).

2. To switch between run levels, use the **`init`** command. For example, to switch to run level 3, enter the command:

   ```
   init 3
   ```

   Here is a short description of the different run levels that are used in Linux:

   – 0 - Halt (Do not set initdefault to this or the server will immediately shut down after finishing the boot process.)

   – 1 - Single user mode

   – 2 - Multi-user, without NFS (the same as 3, if you do not have networking)

   – 3 - Full multi-user mode

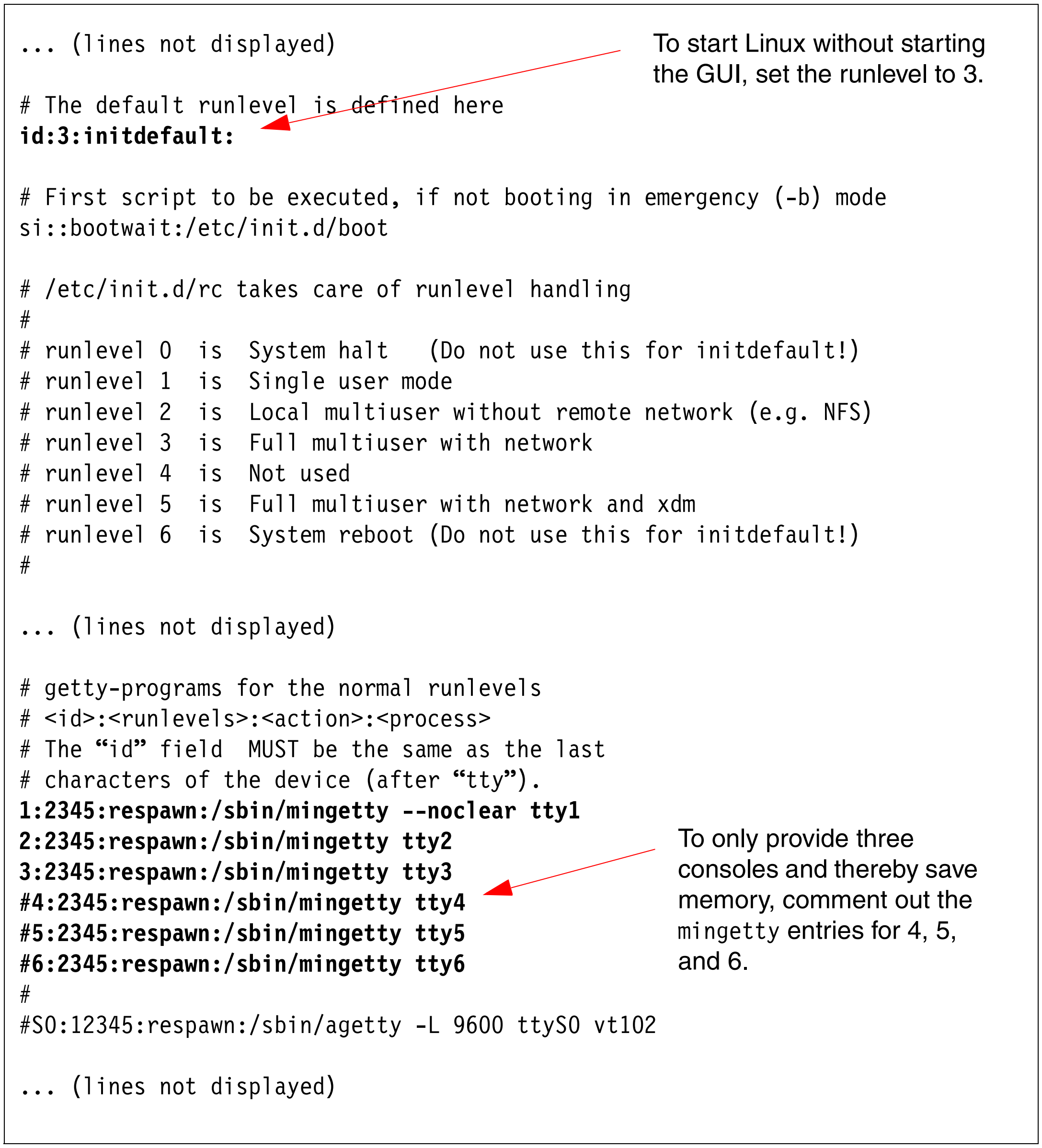
   – 4 - Unused

   – 5 - X11

   – 6 - Reboot (Do not set initdefault to this or the server machine will continuously reboot at startup.)

To set the initial runlevel of a machine at boot, modify the /etc/inittab file as shown in Figure 1-3 on page 5 with the line:

```
id:3:initdefault:
```



```
... (lines not displayed)

# The default runlevel is defined here
id:3: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!)
#

... (lines not displayed)

# getty-programs for the normal runlevels
# <id>:<runlevels>:<action>:<process>
# The “id” field  MUST be the same as the last
# characters of the device (after “tty”).
1:2345:respawn:/sbin/mingetty --noclear tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
#4:2345:respawn:/sbin/mingetty tty4
#5:2345:respawn:/sbin/mingetty tty5
#6:2345:respawn:/sbin/mingetty tty6
#
#S0:12345:respawn:/sbin/agetty -L 9600 ttyS0 vt102

... (lines not displayed)
```

To start Linux without starting the GUI, set the runlevel to 3.

To only provide three consoles and thereby save memory, comment out the `mingetty` entries for 4, 5, and 6.

Figure 1-3 /etc/inittab, modified (only part of the file is displayed)

With SUSE LINUX Enterprise Server 8, this can also be accomplished in by running the **`YaST runlevel`** command and changing the default runlevel. See Figure 1-1 on page 3.

By default, six consoles are saved: F1...F6 are separate consoles. To regain some memory you may wish to limit the number of consoles to three from the original six. To do this, comment out each `mingetty ttyx` line you want to disable. In Figure 1-3, for example, we have limited the consoles to three.

> **Tip:** Even if you have the GUI disabled locally on the server, you can still connect remotely and use the GUI. To do this, use the **`-X`** parameter on the **`ssh`** command.

1.3 Compiling the kernel

Compiling the kernel is not absolutely necessary to improve the performance and will reduce the supportability of the server, but we do recommend that you configure your Linux server to have the latest patches, kernel, and drivers provided from SUSE LINUX and the hardware vendors. There are always new improvements being made available that not only fix bugs, but also improve the performance of the Linux machine.

Restriction: Recompiling the kernel or the addition of kernel modules other than those provided by SUSE voids standard SUSE service and maintenance liabilities. You should contact SUSE LINUX to request an individual service agreement on such a non-standard setup.

Before you begin, you will need to know what hardware is installed in the server. The most important things to know are:

- CPU type
- Amount of memory installed
- SCSI adapter
- RAID controller
- Fibre Channel adapter
- Network adapter
- Video adapter

The more information you have about the hardware that is being used, the easier it will be to configure and compile a new kernel.

We also recommend that you create an emergency boot disk. Before creating the diskette, you will need to know the version of the kernel that is currently installed on the server. One way is to use the following command:

```
uname -a
```

Example 1-1 shows an example of the output of this command.

Example 1-1 Output of uname

```
Linux ibm-dn1ao2berub 2.4.21-215-default #1 Tue Apr 27 16:17:49 UTC 2004 i686
unknown
```

In SUSE LINUX Enterprise Server 8, the easiest way to create a bootdisk is through YaST. From a command line, type **`yast bootfloppy`** and follow the prompts.

Note: In order to use the YaST bootdisk function, you will need the SUSE LINUX Enterprise Server 8 installation CD-ROMs.

We recommend that you test the boot disk before starting to compile the kernel.

A complete discussion of how to compile the kernel is covered in Chapter 4 of the IBM Redpaper *Running the Linux 2.4 Kernel on IBM @server xSeries Servers*, REDP0121, available from:

http://www.redbooks.ibm.com/abstracts/redp0121.html

1.4 Changing kernel parameters

The Linux kernel is the core of the operating system (OS) and is common to all Linux distributions. You can make changes to the kernel by modifying parameters that control the OS. These changes are made on the command line using the **sysctl** command.

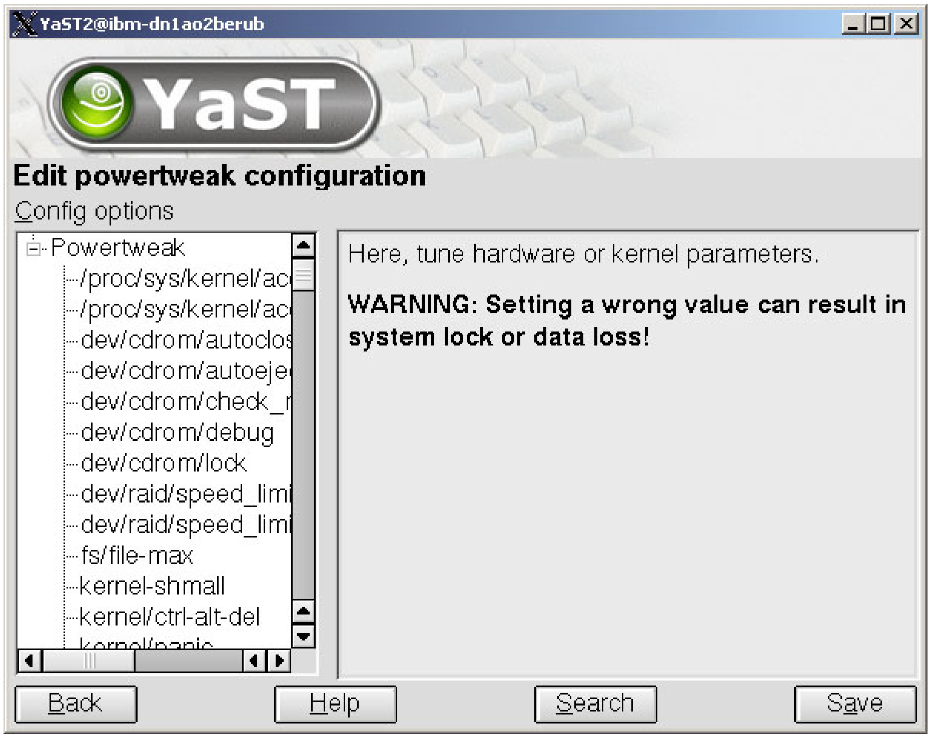
> **Tip:** By default, the kernel includes the necessary module to enable you to make changes using **sysctl** without needing to reboot. However, if you chose to remove this support (during the operating system installation), you will have to reboot Linux before the change will take effect.

In addition, SUSE LINUX offers a graphical method of modifying these sysctl parameters. To launch the powertweak tool, issue the following command:

```
/sbin/yast powertweak
```

For a text-based menu version, use the command:

```
/sbin/yast2 powertweak
```



Edit powertweak configuration

Here, tune hardware or kernel parameters.

WARNING: Setting a wrong value can result in system lock or data loss!

Figure 1-4 SUSE LINUX powertweak

Where the parameters are stored

The kernel parameters that control how the kernel behaves are stored in /proc (and in particular, /proc/sys).

Reading the files in the /proc directory tree provides a simple way to view configuration parameters that are related to the kernel, processes, memory, network and other components. Each process running in the system has a directory in /proc with the process ID (PID) as name. Figure 1-2 on page 8 lists some of the files that contain kernel information.

Table 1-2 Parameter files in /proc

File/directory	Purpose
/proc/loadavg	Information about the load of the server in 1-minute, 5-minute, and 15-minute intervals. The **uptime** command gets information from this file.
/proc/kcore	Contains data to generate a core dump at run time, for kernel debugging purposes. The command to create the core dump is **gdb** as in: **#gdb /usr/src/linux/vmlinux /proc/kcore**
/proc/stat	Kernel statistics as process, swap and disk I/O.
/proc/cpuinfo	Information about the installed CPUs.
/proc/meminfo	Information about memory usage. The **free** command uses this information.
/proc/sys/abi/*	Used to provide support for “foreign” binaries, not native to Linux — those compiled under other UNIX® variants such as SCO UnixWare 7, SCO OpenServer, and SUN Solaris 2. By default, this support is installed, although it can be removed during installation.
/proc/sys/fs/*	Used to increase the number of open files the OS allows and to handle quota.
/proc/sys/kernel/*	For tuning purposes, you can enable hotplug, manipulate shared memory, and specify the maximum number of pid files and level of debug in syslog.
/proc/sys/net/*	Tuning of network in general, IPV4 and IPV6.
/proc/sys/vm/*	Management of cache memory and buffer.

Using the sysctl command

The **sysctl** commands use the names of files in the /proc/sys directory tree as parameters. For example, to modify the shmmax kernel parameter, you can display (using **cat**) and change (using **echo**) the file /proc/sys/kernel/shmmax:

```
#cat /proc/sys/kernel/shmmax
33554432
#echo 33554430 > /proc/sys/kernel/shmmax
#cat /proc/sys/kernel/shmmax
33554430
```

However, using these commands can easily introduce errors, so we recommend that you use the **sysctl** command because it checks the consistency of the data before it makes any change. For example:

```
#sysctl kernel.shmmax
kernel.shmmax = 33554432
#sysctl -w kernel.shmmax=33554430
kernel.shmmax = 33554430
#sysctl kernel.shmmax
kernel.shmmax = 33554430
```

This change to the kernel will stay in effect only until the next reboot. If you want to make the change permanently, you can edit the /etc/sysctl.conf file and add the appropriate command. In our example:

```
kernel.shmmax = 33554439
```

The next time you reboot, the parameter file will be read. You can do the same thing without rebooting by issuing the following command:

```
#sysctl -p
```

1.5 V2.4 kernel parameters

Version 2.4 of the Linux kernel has many parameters that can improve performance for your installation.

Table 1-3 lists the SUSE LINUX parameters that are most relevant to performance. Table 1-4 on page 11 lists relevant parameters that are not normally used in performance tuning.

Table 1-3 SUSE LINUX parameters that are most relevant to performance tuning

Parameter	Description and example of use
kernel.shm-bigpages-per-file	Normally used for tuning database servers. The default is 32768. To calculate a suitable value, take the amount of SGA memory in GB and multiply by 1024. For example: `sysctl -w kernel.shm-bigpages-per-file=16384`
kernel.sched_yield_scale	Enables the dynamic resizing of time slices given to processes. When enabled, the kernel reserves more time slices for busy processes and fewer for idle processes. The parameters kernel.min-timeslice and kernel.max-timeslice are used to specify the range of time slices that the kernel can supply as needed. If disabled, the time slices given to each process are the same. The default is 0 (disabled). Applications such as ERP and Java™ can benefit from enabling this. For real-time applications such as streaming audio and video, leave it disabled. For example: `sysctl -w kernel.sched_yield_scale=1`
kernel.shm-use-bigpages	Enables the use of bigpages (typically for databases). Default is 0 (disabled). For example: `sysctl -w kernel.shm-use-bigpages=1`
net.ipv4.conf.all.hidden	All interface addresses are hidden from ARP broadcasts and will be included in the ARP response of other addresses. Default is 0 (disabled). For example: `sysctl -w net.ipv4.conf.all.hidden=1`
net.ipv4.conf.default.hidden	Enables all interfaces as hidden by default. Default is 0 (disabled). `sysctl -w net.ipv4.conf.default.hidden=1`
net.ipv4.conf.eth0.hidden	Enables only interface eth0 as hidden. Uses the ID of your network card. Default is 0 (disabled). `sysctl -w net.ipv4.conf.eth0.hidden=1`
net.ipv4.ip_conntrack_max	This setting is the number of separate connections that can be tracked. Default is 65536. `sysctl -w net.ipv4.ip_conntrack_max=32768`
net.ipv6.conf.all.mtu	Default maximum for transfer unit on IPV6. Default is 1280. `sysctl -w net.ipv6.conf.all.mtu=9000`
net.ipv6.conf.all.router_solicitation_delay	Determines whether to wait after interface opens before sending router solicitations. Default is 1 (the kernel should wait). For example: `sysctl -w net.ipv6.conf.all.router_solicitation_delay=0`
net.ipv6.conf.all.router_solicitation_interval	Number of seconds to wait between router solicitations. Default is 4 seconds. For example: `sysctl -w net.ipv6.conf.all.router_solicitation_interval=3`
net.ipv6.conf.all.router_solicitations	Number of router solicitations to send until assuming no routers are present. Default is 3. `sysctl -w net.ipv6.conf.all.router_solicitations=2`

Parameter	Description and example of use
net.ipv6.conf.all.temp_prefered_lft	Lifetime preferred in seconds for temporary addresses. Default is 86400 (1 day). `sysctl -w net.ipv6.conf.all.temp_prefered_lft=259200`
net.ipv6.conf.all.temp_valid_lft	Lifetime valid in seconds for temporary address. Default is 604800 (1 week). `sysctl -w net.ipv6.conf.all.temp_valid_lft=302400`
net.ipv6.conf.default.accept_redirects	Accepts redirects sent by a router that works with IPV6, but it cannot set if forwarding is set to enable. Always one or other, it can never set together because it will cause problems in all-IPV6 networks. Default is 1 (enabled). `sysctl -w net.ipv6.conf.default.accept_redirects=0`
net.ipv6.conf.default.autoconf	This automatically generates an address such as "ff81::221:21ff:ae44:2781" on an interface with an L2-MAC Address. Default is 1 (enabled). `sysctl -w net.ipv6.conf.default.autoconf=0`
net.ipv6.conf.default.dad_transmits	Determines whether Duplicate Address Detection (dad) probes are sent. Default is 1 (enabled). `sysctl -w net.ipv6.conf.default.dad_transmits=0`
net.ipv6.conf.default.mtu	Sets the default value for MTU (Maximum Transmission Unit). Default is 1280. `sysctl -w net.ipv6.conf.default.mtu=9000`
net.ipv6.conf.default.regen_max_retry	Number of attempts to try to generate a valid temporary address. Default is 5. `sysctl -w net.ipv6.conf.default.regen_max_retry=3`
net.ipv6.conf.default.router_solicitation_delay	Number in seconds to wait, after interface is brought up, before sending router request. Default is 1 (enabled). `sysctl -w net.ipv6.conf.default.router_solicitation_delay=0`
vm.heap-stack-gap	Enables the heap of memory that is used to store information about status of process and local variables. You should disable this when you need to run a server with JDK; otherwise your software will crash. Default is 1 (enabled). `sysctl -w vm.heap-stack-gap=0`
vm.vm_anon_lru	Allows the VM to always have visibility of anonymous pages. Default is 1 (enabled). `sysctl -w vm.vm_anon_lru=0`
vm.vm_lru_balance_ratio	Balances active and inactive sections of memory. Define the amount of inactive memory that the kernel will rotate. Default is 2. `sysctl -w vm.vm_lru_balance_ratio=3`
vm.vm_mapped_ratio	Controls the pageout rate. Default is 100. `sysctl -w vm.vm_mapped_ratio=90`
vm.vm_passes	Number of attempts that the kernel will try to balance the active and inactive sections of memory. Default is 60. `sysctl -w vm.vm_passes=30`

Parameter	Description and example of use
vm.vm_shmem_swap	Improves performance of applications that use large amounts of non-locked shared memory (such as ERP and database applications) on a server with more than 8 GB of RAM. Default is 0 (disabled). `sysctl -w vm.vm_shmem_swap=1`
vm.vm_vfs_scan_ratio	Proportion of Virtual File System unused caches that will try to be in one VM freeing pass. Default is 6. `sysctl -w vm.vm_vfs_scan_ratio=6`

Table 1-4 SUSE LINUX performance parameters that typically are not used

Parameter	Description and example of use
cpu.0.speed	**Note**: We recommend that you do not use this parameter on a production system. Controls the speed (frequency) of CPU 0. Default is 0, which disables this parameter. Any non-zero value specifies a frequency in Hertz. To set the frequency to 800 MHz: `sysctl -w cpu.0.speed=800000`
cpu.0.speed-max	**Note**: We recommend that you do not use this parameter on a production system. Sets maximum CPU frequency that the processor can reach. This can be used to specify the maximum speed for tools such as SpeedStep. Default is 0, which disables this parameter. Any non-zero value specifies a frequency in Hertz. `sysctl -w cpu.0.speed-max=1000000`
cpu.0.speed-min	**Note**: We recommend that you do not use this parameter on a production system. Sets minimum CPU frequency that the processor can reach. This can be used to specify the maximum speed for tools such as SpeedStep. Default is 0, which disables this parameter. Any non-zero value specifies a frequency in Hertz. `sysctl -w cpu.0.speed-min=900000`
kernel.HZ	**Note**: We recommend that you do not use this parameter on a production system. Some programs (for example, procps) need the HZ parameter set in the kernel. It can be determined by counting jiffies in a second. Default is 100. `sysctl -w kernel.HZ=80`
kernel.max-timeslice	The maximum kernel time granted for a specific process before another process is given kernel time. This value is important if kernel.sched_yield_scale is set to 1 (enabled). Default is 310000. `sysctl -w kernel.max-timeslice=300000`
kernel.min-timeslice	The minimum kernel time granted for a specific process before another process is given kernel time. This value is important if kernel.sched_yield_scale is set to 1 (enabled). Default is 10000. `sysctl -w kernel.min-timeslice=9000`
net.ipv4.conf.lo.hidden	Make the loopback device hidden. Default is 0 (disabled). `sysctl -w net.ipv4.conf.lo.hidden=1`
vm.vm_cache_scan_ratio	This is the fraction of inactive LRU queue during a normal aging round. Default is 6, so the kernel will scan 1/6. `sysctl -w vm.vm_cache_scan_ratio=5`

1.6 V2.6 kernel parameters

Version 2.6 of the Linux kernel has many parameters that can improve performance for your installation. For more information about what is new with the V2.6 kernel, see:

http://www.infoworld.com/infoworld/article/04/01/30/05FElinux_1.html

Note that some V2.4 features were removed from V2.6, including:

- Support for Direct Rendering Manager (DRM)
- Logical Volume Manager 1
- Compress VFAT (CVF) support

In addition, some features are still available in 2.6 but have been deprecated and may be removed in future kernel releases, including the devfs file system. Devfs is replaced by the udev file system. This also means that to tune the I/O scheduler, you should use the file that is exported in the sysfs directory /sys/block/<device>/queue/ioshed instead of elvtune because elvtune uses IOCTLs, which is also deprecated.

Table 1-5 lists the sysctl parameters in the 2.6 kernel that are relevant to performance.

> **Important:** The use of the V2.6 kernel on SUSE LINUX Enterprise Server 8 is not supported by SUSE LINUX.

Table 1-5 V2.6 kernel parameters that impact server performance

Parameter	Description and example of use
kernel.mmap-hugepages-min-mapping	Sets the minimum size for hugepages created with mmap() system call. Default is 256. `sysctl -w kernel.mmap-hugepages-min-mapping=2048`
kernel.mmap-use-hugepages	Enables the use for mmap() system call to create hugepages. Default is 0 (disabled). `sysctl -w kernel.mmap-use-hugepages=1`
kernel.shm-use-hugepages	Enables the use of hugepages. When enabled, pages will be created with standard SYSv shared memory system call. Default is 0 (disable). `sysctl -w kernel.shm-use-hugepages=1`
net.ipv4.conf.all.arp_announce	Defines different restrictions between source IP address and from IP packages in ARP requests sent on interfaces. Default is 0, and valid values are 0, 1, and 2. `sysctl -w net.ipv4.conf.all.arp_announce=2`
net.ipv4.conf.all.arp_ignore	Defines different modes for replies in response to ARP requests. Default is 0. Valid values are 0, 1, 2, 3, and 8. `sysctl -w net.ipv4.conf.all.arp_ignore=8`
net.ipv4.inet_peer_gc_maxtime	Advises the garbage collector how often to pass over the inet peer storage memory pool during low or absent memory pressure. Default is 120, measured in jiffies. `sysctl -w net.ipv4.inet_peer_gc_maxtime=240`
net.ipv4.inet_peer_gc_mintime	Sets the minimum time that the garbage collector can pass cleaning memory. If your server has a heavy load, you should increase this time. Default is 10, measured in jiffies. `sysctl -w net.ipv4.inet_peer_gc_mintime=80`

Parameter	Description and example of use
net.ipv4.inet_peer_maxttl	The maximum time-to-live for the inet peer entries. New entries will expire after this period of time if there is no memory pressure on the pool. Default is 600, measured in jiffies. `sysctl -w net.ipv4.inet_peer_maxttl=500`
net.ipv4.inet_peer_minttl	The minimum time-to-live for inet peer entries. Set to a high-enough value to cover fragment time to live in the reassembling side of fragmented packets. This minimum time must be smaller than net.ipv4.inet_peer_threshold. Default is 120, measured in jiffies. `sysctl -w net.ipv4.inet_peer_minttl=80`
net.ipv4.inet_peer_threshold	Sets the size of inet peer storage. When this limit is nearly reached, peer entries will be thrown away, using the inet_peer_gc_mintime timeout. Default is 65644. `sysctl -w net.ipv4.inet_peer_threshold=65644`
vm.lower_zone_protection	Allows the increase of the amount of protected memory that lower zones receive against allocations that could use higher zones. Default is 0. `sysctl -w vm.lower_zone_protection`
vm.min_free_kbytes	Sets the value of each page's zone to min, low, or high, depending on the machine. Default is 724. For a small machine use 128 KB; for large machines, use the maximum 4096 KB. `sysctl -w vm.min_free_kbytes=512`
vm.nr_hugepages	Try to allocate or deallocate hugepages dynamically. The success of this operation depends on the amount of memory that is present in the system for this operation. Default is 0. `sysctl -w vm.nr_hugepages=10`

Table 1-6 V2.6 kernel performance parameters that typically are not used

Parameter	Description and example of use
kernel.panic_on_oops	Enable kernel detection and handling of any process that causes a crash and calls the panic() function at the end. The kernel.panic parameter must also be set to 1. Default is 1 (enable). `sysctl -w kernel.panic_on_oops=0`
kernel.pid_max	Determines the maximum pid that a process can allocate. Default is 32768. `sysctl -w kernel.pid_max=65536`
net.ipv4.tcp_tw_recycle	The main states of TCP connection are ESTABLISHED, TIME_WAIT, and CLOSED, so it enables the fast recycling function of TIME-WAIT sockets. Default is 0. `sysctl -w net.ipv4.tcp_tw_recycle=10`
net.ipv4.tcp_westwood	Enables TCP Westwood+ congestion control algorithm that optimizes the performance of TCP congestion control. Default is 0. `sysctl -w net.ipv4.tcp_westwood=1`
net.ipv6.conf.all.max_addresses	Sets maximum number of addresses per interface. Default is 0. `sysctl -w net.ipv6.conf.all.max_addresses=10`
vm.overcommit_ratio	Percentage of memory that is allowed for overcommit. Default is 50. `sysctl -w vm.overcommit_ratio=17`

Parameter	Description and example of use
vm.swappiness	Determines how likely the VM subsystem is to swap to disk. Default is 60 and typically is sufficient. Valid values are 0 to 100. `sysctl -w vm.swappiness=50`

1.7 Tuning the processor subsystem

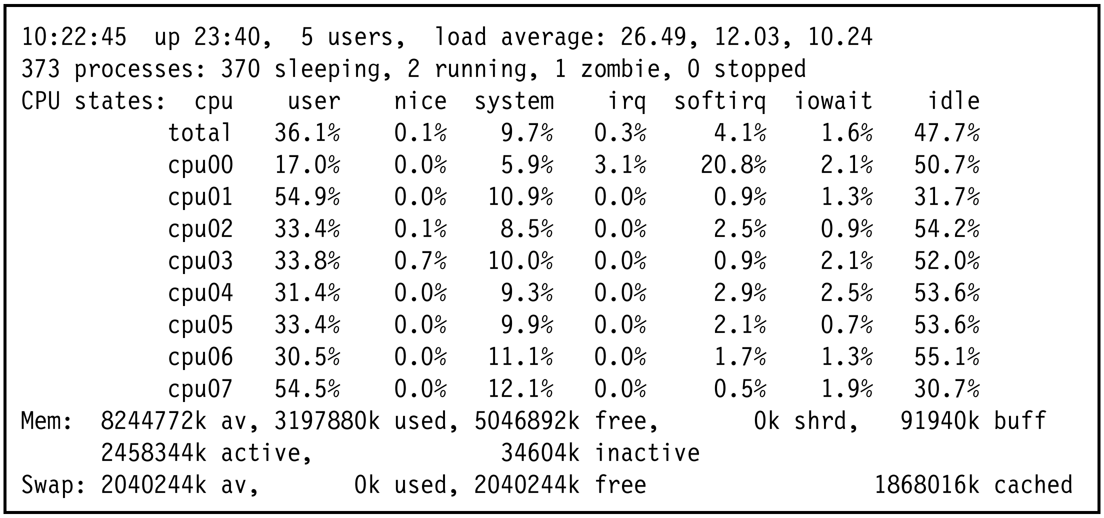
The CPU is one of the most important hardware subsystems for servers whose primary role is that of an application or database server. However, in these systems, the CPU is often the source of performance bottlenecks.

For tweaking of processor tuning parameters, refer to 1.5, “V2.4 kernel parameters” on page 9 and 1.6, “V2.6 kernel parameters” on page 12.

On high-end servers with Xeon processors, you may wish to enable or disable Hyper-Threading.

Hyper-Threading is a way of virtualizing each physical processor as two processors to the OS. This is supported under SUSE LINUX Enterprise Server 8 and SUSE LINUX Enterprise Server 9. By virtualizing the processor you can execute two threads or processes at a time (also know as *thread-level parallelism*). By having your OS and software designed to take advantage of this technology you can gain significant increases in performance without needing an increase in clock speed.

For example, if you have Hyper-Threading enabled on a 4-way server, monitoring tools such as **`top`** will display eight processors. See Figure 1-5.



```
10:22:45  up 23:40,  5 users,  load average: 26.49, 12.03, 10.24
373 processes: 370 sleeping, 2 running, 1 zombie, 0 stopped
CPU states:  cpu    user    nice  system    irq  softirq  iowait    idle
           total   36.1%    0.1%    9.7%   0.3%     4.1%    1.6%   47.7%
           cpu00   17.0%    0.0%    5.9%   3.1%    20.8%    2.1%   50.7%
           cpu01   54.9%    0.0%   10.9%   0.0%     0.9%    1.3%   31.7%
           cpu02   33.4%    0.1%    8.5%   0.0%     2.5%    0.9%   54.2%
           cpu03   33.8%    0.7%   10.0%   0.0%     0.9%    2.1%   52.0%
           cpu04   31.4%    0.0%    9.3%   0.0%     2.9%    2.5%   53.6%
           cpu05   33.4%    0.0%    9.9%   0.0%     2.1%    0.7%   53.6%
           cpu06   30.5%    0.0%   11.1%   0.0%     1.7%    1.3%   55.1%
           cpu07   54.5%    0.0%   12.1%   0.0%     0.5%    1.9%   30.7%
Mem:  8244772k av, 3197880k used, 5046892k free,       0k shrd,   91940k buff
      2458344k active,            34604k inactive
Swap: 2040244k av,       0k used, 2040244k free                 1868016k cached
```

Figure 1-5 Output of top on a four-way server with Hyper-Threading enabled

Note the following with respect to Hyper-Threading:

- SMP-based kernels are required to support Hyper-Threading.
- The more CPUs installed in a server, the less benefit Hyper-Threading has on performance. On servers that are CPU-bound, expect, at most, the following performance gains:
  - Two physical processors: 15-25% performance gain
  - Four physical processors: 1-13% gain

  - Eight physical processors: 0-5% gain

For more information about Hyper-Threading, see:

http://www.intel.com/business/bss/products/hyperthreading/server/

EM64T is a 64-bit extension to Intel® IA-32 processors, which means that the processors are capable of addressing more memory and can support new 64-bit applications while remaining fully compatible with all existing 32-bit applications. Support for this new processor is in SUSE LINUX Enterprise Server 9.

For more information about EM64T, see:

http://www.intel.com/technology/64bitextensions/

Selecting the correct kernel

SUSE LINUX Enterprise Server 8 includes several kernel packages, as listed in Table 1-7. It is important for performance reasons that you select the most appropriate kernel for your system.

Table 1-7 Available kernels within the distribution

Kernel type	Description
SMP	Kernel has support for SMP and Hyper-Threaded machines.
Standard	Single processor machines.

1.8 Tuning the memory subsystem

Tuning the memory subsystem is a difficult task that requires constant monitoring to ensure that changes do not negatively affect other subsystems in the server. If you do choose to modify the virtual memory parameters (in /proc/sys/vm), we recommend that you change only one parameter at a time and monitor how the server performs.

Tuning tasks on these VM parameters includes:

- Configuring how the Linux kernel will flush dirty buffers to disk. Disk buffers are used to cache data that is stored on disks, which are very slow compared with RAM. So, if the server uses this kind of memory, it can create serious problems with performance. Whenever a buffer becomes sufficiently dirty, use:

  ```
  sysctl -w vm.bdflush="30 500 0 0 500 3000 60 20 0"
  ```

  vm.bdflush has nine parameters, but we recommend that you change only three of them:

  - Parameter 1 is nfract, the maximum percentage of buffer that the bdflush daemon will allow before queuing the buffer to be written to disk.
  - Parameter 2 is ndirty, the maximum number of buffers that bdflush will flush at once. If this value is very large, the bdflush daemon will need more time to finish the update to disk.
  - Parameter 7 is nfract_sync, the maximum percentage of the buffer cache that is dirty before a synchronous flush occurs.

  Other parameters can be left as their default values. A further discussion of bdflush can be found in “Setting bdflush” on page 22.

- Configuring the kswapd daemon to specify how many pages of memory are paged out by Linux.

  ```
  sysctl -w vm.kswapd="1024 32 64"
  ```

  The three parameters are as follows:

  - tries_base is 4 times the number of pages that the kernel swaps in one pass. On a system with a lot of swapping, increasing the number may improve performance.
  - tries_min is the minimum number of pages that kswapd swaps out each time the daemon is called.
  - swap_cluster is the number of pages that kswapd writes at once. A smaller number increases the number of disk I/Os performed, but a larger number may also have a negative impact on the request queue.

If you do make changes, check their impact using tools such as **vmstat**.

Other relevant VM parameters that may improve performance include:

- buffermem
- freepages
- overcommit_memory
- page-cluster
- pagecache
- pagetable_cache

1.9 Tuning the file system

Ultimately, all data must be retrieved from and stored to disk. Disk accesses are usually measured in milliseconds and are thousands of times slower than other components (such as memory or PCI operations, which are measured in nanoseconds or microseconds). The Linux file system is the method by which data is stored and managed on the disks.

Many different file systems are available for Linux that differ in performance and scalability. Besides storing and managing data on the disks, file systems are also responsible for guaranteeing data integrity. The newer Linux distributions include *journaling* file systems as part of their default installation. Journaling, or logging, prevents data inconsistency in case of a system crash. All modifications to the file system metadata have been maintained in a separate journal or log and can be applied after a system crash to bring it back to its consistent state. Journaling also improves recovery time, because there is no need to perform file system checks at system reboot.

As with other aspects of computing, you will find that there is a trade-off between performance and integrity. However, as Linux servers make their way into corporate data centers and enterprise environments, requirements such as high availability can be addressed.

In this section we cover the default file system as well as additional file systems that are available on SUSE LINUX Enterprise Server and some simple ways to improve their performance.

1.9.1 Hardware considerations before installing Linux

Minimum requirements for CPU speed and memory are well documented for current Linux distributions. Those instructions also provide guidance for the minimum disk space that is required to complete the installation. However, they fall short on how to initially set up the disk subsystem. Because Linux servers cover a vast assortment of work environments as server consolidation makes its impact in data centers, one of the first questions to answer is: What is the function of the server being installed?

A server's disk subsystems can be a major component of overall system performance. Understanding the function of the server is key to determining whether the I/O subsystem will be a direct impact to performance.

Examples of servers where disk I/O is most important:

- A file and print server must move data quickly between users and disk subsystems. Because the purpose of a file server is to deliver files to the client, the server must initially read all data from a disk.
- A database server's ultimate goal is to search and retrieve data from a repository on the disk. Even with sufficient memory, most database servers will perform large amounts of disk I/O to bring data records into memory and flush modified data to disk.

Examples of servers where disk I/O is not the most important subsystem:

- An e-mail server acts as a repository and router for electronic mail and tends to generate a heavy communication load. Networking is more important for this type of server.
- A Web server that is responsible for hosting Web pages (static, dynamic, or both) benefits from a well-tuned network and memory subsystem.

Disk technology selection

Besides understanding the function of the server, you must also understand the size of the deployment that the installation will have to serve. Current disk subsystem technologies were designed with size of deployment in mind. Table 1-8 briefly describes the disk technologies that are currently available with the IBM xSeries servers.

Table 1-8 Current disk technologies

Technology	Cost	Function	Limitations and capabilities
EIDE	Lowest cost	Direct-attached storage; for example, low-end servers, local storage (x305)	An extension of IDE that is used for connecting internal storage. Maximum: two drives per EIDE controller.
SCSI	Low cost	Direct-attached storage; for example, mid-range to high-end server with local storage (x346, x365)	Although the standard for more than 10 years, current I/O demands on high-end servers have stretched the capabilities of SCSI. Limitations include cable lengths, transfer speeds, maximum number of attached drives, and limits on number of systems that can actively access devices on one SCSI bus, affecting clustering capabilities.
Serial ATA (SATA)	Low cost	Midrange data-storage applications	Generally available since late 2002, this new standard in HDD/system board interface is the follow-on technology to EIDE. With its point-to-point protocol, scalability improves as each drive has a dedicated channel. Sequential disk access is comparable to SCSI but random access is less efficient. RAID functionality is also available.
iSCSI	Medium cost	Mid-end storage; for example, File/Web server	Became an RFC recently. Currently being targeted toward mid-end storage and remote booting. Primary benefits are savings in infrastructure cost and diskless servers. It also provides the scalability and reliability associated with TCP/IP/Ethernet. High latency of TCP/IP limits performance.

Technology	Cost	Function	Limitations and capabilities
Fibre Channel	High cost	Enterprise Storage; for example, databases	Provides low latency and high throughput capabilities and removes the limitations of SCSI by providing cable distances of up to 10 km with fiber optic links; 2 Gbps transfer rate, redundant paths to storage to improve reliability; in theory can connect up to 16 million devices; in loop topologies, up to 127 storage devices or servers can share the same Fibre Channel connection allowing implementations of large clusters.

For additional information about available IBM storage solutions, visit:

`http://www.ibm.com/storage`

Number of drives

The number of disk drives significantly affects performance because each drive contributes to total system throughput. Capacity requirements are often the only consideration that is used to determine the number of disk drives that are configured in a server. Throughput requirements are usually not well understood or are completely ignored. The key to a good performing disk subsystem depends on maximizing the number of read-write heads that can service I/O requests.

With RAID (redundant array of independent disks) technology, you can spread the I/O over multiple spindles. There are two options for implementing RAID in a Linux environment: software RAID or hardware RAID. Unless your server hardware comes standard with hardware RAID, you may want to start with the software RAID options that come with the Linux distributions; if a need arises, you can grow into the more efficient hardware RAID solutions.

Software RAID in the 2.4 Linux kernel distributions is implemented through the md device driver layer. This driver implementation is device-independent and therefore is flexible in allowing many types of disk storage such as EIDE or SCSI to be configured as a RAID array. Supported software RAID levels are RAID-0 (striping), RAID-1 (mirroring), and RAID-5 (striping with parity) and can be completed as part of the initial installation or through the **`mdadm`** tool set.

If it is necessary to implement a hardware RAID array, you will need a RAID controller for your system. In this case the disk subsystem consists of the physical hard disks and the controller. IBM offers a complete product line of controllers, as shown in Table 1-9.

Table 1-9 Available IBM RAID controllers

Storage controller	Product name	Features
ServeRAID™ Family	ServeRAID 7T	Entry-level 4-port SATA controller, supports RAID level 0,1,10, and 5.
	ServeRAID 6M	2-channel Ultra320 SCSI with 14 disk drives per channel, supports RAID levels 0,1,10,1E, 5, 50, and 5EE.
	ServeRAID 6I	Cost-effective “zero channel” using the onboard SCSI chipset, supports standard RAID levels 0,00,1,10,5,50, and IBM-exclusive 1E, 5EE, and1E0.

Storage controller	Product name	Features
FAStT	FAStT100	Entry-level storage server with support for up to 56 SATA drives and dual active 2 GB RAID controllers.
	FAStT 200	Compact 3U size with full integrated Fibre Channel technology supporting up to 10 internal FC disk drives and a max of 66 with additional external enclosures available in both single and dual (HA) controller models.
	FAStT 600	Models include single and dual controller supporting from 14 FC drives up to 112 FC or SATA disks with Turbo model. Turbo model also provides a 2 GB cache.
	FAStT 700	Dual active RAID controllers, transfer rates of 2 Gbps and support for up to 224 drives for a maximum physical capacity of 32 TB; 2 GB battery-backed controller cache.
	FAStT 900	Dual active 2 GB RAID controllers, up to 795 MBps throughput and support for up to 32 TB of FC disk storage or 56 TB of SATA storage; 2 GB battery-packed controller cache can support high-performance applications such as OLTP and data mining.

Tip: In general, adding drives is one of the most effective changes that can be made to improve server performance.

For additional, in-depth coverage of the available IBM storage solutions, see:

- *IBM TotalStorage Disk Solutions for xSeries*, SG24-6874

  http://www.redbooks.ibm.com/abstracts/sg246874.html

- *IBM @server xSeries ServeRAID Technology*

  http://www.pc.ibm.com/ww/eserver/xseries/scsi_raid.html

1.9.2 ReiserFS, the default SUSE LINUX file system

The default file system on a SUSE installation since SUSE LINUX 7.1 has been ResierFS, developed by Hans Reiser. From its initial design, key performance aspects have included:

- Journaling designed into the file system from the beginning improves reliability and recovery.
- Provides faster access through the use of balanced tree data structures that allows for storing both content data and security metadata.
- Efficient use of disk space because, unlike other file systems, this file system does not rely on block sizes.

The current version of ReiserFS that is installed with SUSE LINUX Enterprise Server 8 is V3.6. There is work underway on delivering the next release, Reiser4. The new Reiser4 file system is expected to deliver an unbreakable file system by eliminating corruption with the implementation of an “atomic” file system where I/O is guaranteed to complete, a 2x to 5x speed improvement by implementing new access algorithms, and ease of third-party upgrades without reformatting, through the use of plug-ins.

1.9.3 Other journaling file systems

Here is a sample of current file systems and their features for the Linux Kernel 2.4:

- Ext3 is a based on incremental modifications to the Linux standard ext2 file system.
- ReiserFS was designed and implemented as a journaling file system from the beginning by Hans Reiser.
- JFS was first designed by IBM for OS/2® WARP® Server, and then its code base was used for AIX® JFS2. It has been available since Linux Kernel 2.4.20 and was added to 2.5.6.
- XFS was first introduced by SGI on its IRIX 64-bit systems. It has been available since Linux Kernel 2.4.x and added into 2.5.3.

Note: During installation, SUSE LINUX Enterprise Server 8 only offers the option of installing ext2, ext3, JFS, and XFS file systems.

1.9.4 File system tuning in the Linux kernel

Out-of-the-box settings for the default file systems may be adequate for most environments. However, here are a few a pointers to help improve overall disk performance.

Access time updates

The Linux file system keeps records of when files are created, updated, and accessed. Default operations include updating the last-time-read attribute for files during reads and writes to files. Because writing is an expensive operation, eliminating unnecessary I/O can lead to overall improved performance.

Mounting file systems with the noatime option eliminates the inode access times from being updated. If file update times are not critical to your implementation, as in a Web-serving environment, an end user can choose to mount file systems with the noatime flag in the /etc/fstab file as shown in Example 1-2.

Example 1-2 Update /etc/fstab file with noatime option set on mounted file systems

```
/dev/sdb1 /mountlocation ext3 defaults,noatime 1 2
```

It is generally a good idea to have a separate /var partition and mount it with the noatime option.

Tune the elevator algorithm

The disk I/O elevator algorithm was introduced as a feature in the Version 2.4 kernel. It enables the user to tune the algorithm that schedules block I/O by controlling the amount of time an I/O request remains on the queue before being serviced. This is accomplished by adjusting the read and write values of the elevator algorithm. By increasing latency times (that is, larger values for read, write, or both), I/O requests stay on the queue for a longer period of time, giving the I/O scheduler the opportunity to coalesce these requests to perform more efficient I/O and increase throughput.

If your Linux server is in an environment with large amounts of disk I/O, finding the right balance between throughput and latency may be beneficial. Linux file systems are implemented as block devices, so improving how often those blocks are read and written can improve file system performance. As a guideline, heavy I/O servers benefit from smaller caches, prompt flushes, and a balanced high-latency read to write.

As with other system tuning, tuning the elevator algorithm is an iterative process. You want to baseline current performance, make changes, and then be able to measure the effect of those changes. Example 1-3 shows how the **/sbin/elvtune** command is used to first show the current settings and then change the values for the read and write queues.

If any changes are made, be sure that the **/sbin/elvtune** call is added to the /etc/init.d/boot.local file to make it a persistent change between system boots.

Example 1-3 Finding current defaults for your installation and changing them

```
[root@x232 root]# elvtune /dev/sda

/dev/sda elevator ID            2
        read_latency:           2048
        write_latency:          8192
        max_bomb_segments:      6

[root@x232 root]# elvtune -r 1024 -w 2048 /dev/sda

/dev/sda elevator ID            2
        read_latency:           1024
        write_latency:          2048
        max_bomb_segments:      6
```

> **Note:** As of this publication, documentation for Version 2.6 of the Linux kernels states that the **elvtune** command is obsolete and mentions that tuning on newer kernels can be accomplished through the /sys/block structure.

Select the journaling mode of an ext3 file system

Three different journaling options in the Ext3 file system can be set with the data option in the **mount** command:

- data=journal

  This journaling option provides the highest form of data consistency by causing both file data and metadata to be journalled. It is also has the higher performance overhead.

- data=ordered (default)

  In this mode only metadata is written. However, file data is guaranteed to be written first. This is the default setting.

- data=writeback

  This journaling option provides the fastest access to the data at the expense of data consistency. The data is guaranteed to be consistent as the metadata is still being logged. However, no special handling of actual file data is done and this may lead to old data appearing in files after a system crash.

There are three ways to change the journaling mode on a file system:

- When executing the **mount** command:

  ```
  mount -o data=writeback /dev/sdb1 /mnt/mountpoint
  ```

  - /dev/sdb1 is the file system being mounted.

- Including it in the options section of the /etc/fstab file:

  ```
  /dev/sdb1 /testfs ext3 defaults,journal=writeback 0 0
  ```

- If you want to modify the default **data=ordered** option on the root partition, make the change to the /etc/fstab file listed above, and then execute the **mkinitrd** command to scan the changes in the /etc/fstab file and create a new image. Update grub or lilo to point to the new image.

For more information about Ext3, see:

http://www.redhat.com/support/wpapers/redhat/ext3/

Tuning ReiserFS

One of the strengths of the ReiserFS is its support for a large number of small files. Instead of using the traditional block structure of other Linux file systems, ReiserFS uses a tree structure that has the capability of storing the actual contents of small files or the “tails” of those that are larger in the access tree itself. This file system does not use fixed block sizes, so only the space that is needed to store a file is consumed, leading to less wasted space.

There is an option when mounting a ReiserFS file system that improves performance but at the expense of space. When mounting a ReiserFS, one can disable this “tail packing” option by specifying **notail** so that the file system performs a little faster but uses more disk space.

Example 1-4 Example of mounting a ReiserFS file system with the notail option

```
/dev/sdb1 /testfs resierfs notail 0 0
```

Setting bdflush

There is tuning in the virtual memory subsystem that can help improve overall file system performance. The bdflush kernel daemon is responsible for making sure that dirty buffers, any modified data that currently resides only in the volatile system memory, are committed to disk. By modifying the /proc/sys/vm/bdflush parameters, one can modify the writing-to-disk rate, possibly avoiding disk contention problems. Changes in the /proc system will take effect immediately but will be reset at boot time. To make changes permanent, include the **echo** command in the /etc/rc.d/rc.local file.

The nine parameters in the /proc/sys/vm/bdflush of 2.4 Linux kernels are:

nfract Maximum percentage of dirty buffers in the buffer cache. The higher the value, the longer the write to the disk will be postponed. When available memory is in short supply, large amounts of I/O will have to be processed. To spread I/O out evenly, keep this a low value.

ndirty Maximum number of dirty buffers that the bdflush process can write to disk at one time. A large value results in I/O occurring in bursts, and a small value may lead to memory shortages if the bdflush daemon is not executed enough.

dummy2 Unused (formerly nrefill).

dummy3 Unused.

interval Minimum rate at which kupdate will wake and flush. Default is 5 seconds, with a minimum value of zero seconds and a maximum of 600 seconds.

age_buffer Maximum time the OS will wait before writing buffer cache to disk. Default is 30 seconds, with minimum of 1 second and maximum of 6000 seconds.

nfract_sync Percentage of dirty buffers to activate bdflush synchronously. Default is 60%.

nfract_stop Percentage of dirty buffers to stop bdflush. Default is 20%.

dummy5 Unused.

Example 1-5 Modifying the bdflush parameters in the kernel

```
echo 30 500 0 0 500 30000 60 20 0 > /proc/sys/vm/bdflush
```

Tuning IDE drives

If your server is equipped with an IDE drive, you can use the **hdparam** command to set hard drive performance settings:

```
/sbin/hdparam
```

Some common options:

/sbin/hdparm -c1 /dev/hda Turns on 32-bit I/O on the PCI bus (or /dev/hdb or /dev/hdc, where each IDE block device is defined as /dev/hdx).

/sbin/hdparm -u1 /dev/hda Unmasks other interrupts.

/sbin/hdparm -A /dev/hda Can be used to increase read-ahead when dealing with large sequential files.

/sbin/hdparm -d1 /dev/hda Enables DMA.

/sbin/hdparm -d1 -X66 /dev/hda Enables Ultra DMA transfers.

The IBM @server BladeCenter comes with an internal EIDE drive. By default, the drive has DMA access enabled but write caching disabled. Example 1-6 shows how to set the write caching and its effect on the buffer-cache reads.

Example 1-6 Example use of hdparm on an IBM BladeCenter to turn on write cache

```
[root@blade2 root]# hdparm -Tt /dev/hda

/dev/hda:
 Timing buffer-cache reads:   1884 MB in  2.00 seconds = 942.00 MB/sec
 Timing buffered disk reads:   76 MB in  3.03 seconds =  25.08 MB/sec
[root@blade2 root]# hdparm -W 1 /dev/hda

/dev/hda:
 setting drive write-caching to 1 (on)
[root@blade2 root]# hdparm -Tt /dev/hda

/dev/hda:
 Timing buffer-cache reads:   1932 MB in  2.00 seconds = 966.00 MB/sec
 Timing buffered disk reads:   76 MB in  3.00 seconds =  25.33 MB/sec
[root@blade2 root]#
[root@blade2 root]# hdparm /dev/hda
```

We recommend that you take baseline performance measurements for your current settings before making any changes. In order to test the effects of changing these settings, one can use the command itself invoked as **hdparm -tT /dev/hda** to see the effects on the speed of reading directly from the buffer cache with no disk accesses (-T) or displaying the speed of reading through the buffer cache without any prior caching of data (-t).

Tagged command queuing (TCQ) for SCSI drives

Tagged command queueing (TCQ), first introduced in the SCSI-2 standard, is a method by which commands arriving at the SCSI drive are tagged and reordered while in the queue. This implementation can increase I/O performance in server environments that have a heavy, random workload by reordering the requests to optimize the position of the drive head.

Recently, this method of queueing and reordering pending I/O requests has been extended to IDE drives and is referred to as ATA TCQ or legacy TCQ and Native Command Queuing (NCQ) in the SATA II specification.

Some IBM xSeries servers include the integrated Adaptec AIC-7xxx SCSI controller. By executing cat /proc/scsi/aic7xxx/0, you can check the current TCQ settings in effect. See the explanation in /usr/src/linux-2.4/drivers/scsi/README.aic7xxx for a detailed description of how to change the default SCSI driver settings.

It is not necessary to recompile the kernel to try different settings. You can specify a parameter aic7xxx=global_tag_depth:*xx* by adding a line in /etc/modules.conf, as shown in Example 1-7.

Example 1-7 Setting TCQ option on a server with an Adaptec aic7xxx SCSI card

```
Edit the /etc/modules.conf file to include
options aic7xxx aic7xxx=verbose.global_tag_depth:16
```

Note: If you make a change to /etc/modules.conf that involves a module in initrd, it will require a new image by executing **`mkinitrd`**.

Block sizes

The block size, the smallest amount of data that can be read or written to a drive, can have a direct impact on a server performance. As a guideline, if your server is handling many small files, a smaller block size will be more efficient. If your server is dedicated to handling large files then a larger block size may improve performance. Block sizes cannot be changed on the fly on existing file systems, and only a reformat will modify the current block size.

When a hardware RAID solution is being used, careful consideration must be given to the *stripe size* of the array (or *segment* in the case of fibre channel). The *stripe-unit size* is the granularity at which data is stored on one drive of the array before subsequent data is stored on the next drive of the array. Selecting the correct stripe size is a matter of understanding the predominant request size performed by a particular application.

As a general rule, streaming or sequential content benefits from large stripe sizes by reducing disk head seek time and improving throughput, but the more random type of activity, such as that found in databases, performs better with a stripe size that is equivalent to the record size.

SUSE LINUX Enterprise Server 8 with ReiserFS allows only block sizes of 4 K at this time.

Guidelines for setting up partitions

A partition is a contiguous set of blocks on a drive that are treated as if they were independent disks. The default installation of SUSE LINUX Enterprise Server 8 creates a very monolithic install with only three partitions:

- A swap partition (automatically set to 2x RAM or 2 GB, whichever is larger)
- A small boot partition, /boot (for example, 100 MB)
- All remaining space dedicated to /

There is a great deal of debate in Linux circles about the optimal disk partition. A single root partition method may lead to problems in the future if you decide to redefine the partitions because of new or updated requirements. On the other hand, too many partitions can lead to a file system management problem. During the installation process, Linux distributions permit you to create a multi-partition layout.

There are benefits to running Linux on a multi-partitioned disk:

- Improved security with finer granularity on file system attributes.

  For example, the /var and /tmp partitions are created with attributes that permit very easy access for all users and processes on the system and are susceptible to malicious access. By isolating these partitions to separate disks, you can reduce the impact on system availability if these partitions need to be rebuilt or recovered.

- Improved data integrity, as loss of data with a disk crash would be isolated to the affected partition.

  For example, if there is no RAID implementation on the system (software or hardware) and the server suffers a disk crash, only those partitions on that bad disk would have to be repaired or recovered.

- New installation and upgrades can be done without affecting other more static partitions.

  For example, if the /home file system has not been separated to another partition, it will be overwritten during an operating system upgrade, losing all user files stored on it.

- More efficient backup process.

  Partition layouts must be designed with backup tools in mind. It is important to understand whether backup tools operate on partition boundaries or on a more granular level like file systems.

Table 1-10 lists some of the partitions that you may want to consider separating out from root to provide more flexibility and better performance in your environment.

Table 1-10 Linux partitions and server environments

Partition	Contents and possible server environments
/home	A *file server environment* would benefit from separating out /home to its own partition. This is the home directory for all users on the system, if there are no disk quotas implemented, so separating this directory should isolate a user's runaway consumption of disk space.
/tmp	If you are running a *high-performance computing environment*, large amounts of temporary space are needed during compute time, and then released upon completion.
/usr	This is where the *kernel source tree* and Linux *documentation* (as well as most executable binaries) are located. The /usr/local directory stores the executables that need to be accessed by all users on the system and is a good location to store custom scripts developed for your environment. If it is separated to its own partition, the files will not need to be reinstalled during an upgrade or re-install by simply choosing not to have the partition reformatted.
/var	The /var partition is important in *mail, Web, and print server environments* as it contains the log files for these environments as well as the overall system log. Chronic messages can flood and fill this partition. If this occurs and the partition is not separate from the /, service interruptions are possible. Depending on the environment, further separation of this partition is possible by separating out /var/spool/mail for a mail server or /var/log for system logs.
/opt	The installation of some third-party software products, such as Oracle's *database server***,** default to this partition. If not separate, the installation will continue under / and, if there is not enough space allocated, may fail.

For a more detailed and in-depth understanding of how Linux distributions handle file system standards, see the File System Hierarchy project's home page:

http://www.pathname.com/fhs

1.9.5 The swap partition

The swap device is used when physical RAM is fully in use and the system needs additional memory. When there is no free memory available on the system, it begins paging the least-used data from memory to the swap areas on the disks. The initial swap partition is created during the Linux installation process with current guidelines stating that the size of the swap partition should be two times physical RAM. Linux kernels 2.4 and beyond support swap sizes up to 24 GB per partition with an 8 TB theoretical maximum for 32-bit systems. Swap partitions should reside on separate disks.

If more memory is added to the server after the initial installation, additional swap space must be configured. There are two ways to configure additional swap after the initial install:

- A free partition on the disk can be created as a swap partition. This can be difficult if the disk subsystem has no free space available. In that case, a swap file can be created.
- If there is a choice, the preferred option is to create additional swap partitions. There is a performance benefit because I/O to the swap partitions bypass the file system and all of the overhead involved in writing to a file.

Another way to improve the performance of swap partitions or files is to create multiple swap areas. Linux can take advantage of multiple swap partitions or files and perform the reads and writes in parallel to the disks. After creating the additional swap partitions or files, the /etc/fstab file will contain such entries as those shown in Example 1-8.

Example 1-8 /etc/fstab file

```
/dev/sda2               swap                    swap    sw              0 0
/dev/sdb2               swap                    swap    sw              0 0
/dev/sdc2               swap                    swap    sw              0 0
/dev/sdd2               swap                    swap    sw              0 0
```

Under normal circumstances, Linux would use the /dev/sda2 swap partition first, then /dev/sdb2, and so on, until it had allocated enough swapping space. This means that perhaps only the first partition, /dev/sda2, will be used if there is no need for a large swap space.

Spreading the data over all available swap partitions improves performance because all read/write requests are performed simultaneously to all selected partitions. If you change the file as shown in Example 1-9, you will assign a higher priority level to the first three partitions.

Example 1-9 Modified /ertc/fstab to make parallel swap partitions

```
/dev/sda2               swap                    swap    sw,pri=3        0 0
/dev/sdb2               swap                    swap    sw,pri=3        0 0
/dev/sdc2               swap                    swap    sw,pri=3        0 0
/dev/sdd2               swap                    swap    sw,pri=1        0 0
```

Swap partitions are used from the highest priority to the lowest (where 32767 is the highest and 0 is the lowest). Giving the same priority to the first three disks causes the data to be written to all three disks; the system does not wait until the first swap partition is full before it starts to write on the next partition. The system uses the first three partitions in parallel and performance generally improves.

The fourth partition is used if additional space is needed for swapping after the first three are completely filled up. It is also possible to give all partitions the same priority to stripe the data over all partitions, but if one drive is slower than the others, performance would decrease. A general rule is that the swap partitions should be on the fastest drives available.

Note: The swap space is not a replacement for RAM as it is stored on physical drives that have a significantly slower access time than memory.

1.10 Tuning the network subsystem

The network subsystem should be tuned when the OS is first installed as well as when there is a perceived bottleneck in the network subsystem. A problem here can affect other subsystems: for example, CPU utilization can be affected significantly, especially when block sizes are too small, and memory use can increase if there is an excessive number of TCP connections.

Preventing a decrease in performance

The following **sysctl** commands are used primarily to change security settings, but they also have the side effect of preventing a decrease in network performance. These commands are changes to the default values in SUSE LINUX.

- Disabling the following parameters will prevent a hacker from using a spoofing attack against the IP address of the server:

```
sysctl -w net.ipv4.conf.eth0.accept_source_route=0
sysctl -w net.ipv4.conf.lo.accept_source_route=0
sysctl -w net.ipv4.conf.default.accept_source_route=0
sysctl -w net.ipv4.conf.all.accept_source_route=0
```

- These commands configure the server to ignore redirects from machines that are listed as gateways. Redirect can be used to perform attacks, so we only want to allow them from trusted sources:

```
sysctl -w net.ipv4.conf.eth0.secure_redirects=1
sysctl -w net.ipv4.conf.lo.secure_redirects=1
sysctl -w net.ipv4.conf.default.secure_redirects=1
sysctl -w net.ipv4.conf.all.secure_redirects=1
```

  In addition, you could allow the interface to accept or not accept any ICMP redirects. The ICMP redirect is a mechanism for routers to convey routing information to hosts. For example, the gateway can send a redirect message to a host when the gateway receives an Internet datagram from a host on a network to which the gateway is attached. The gateway checks the routing table to get the address of the next gateway, and the second gateway will route the datagram's Internet to destination on network. Disable these redirects using the following commands:

```
sysctl -w net.ipv4.conf.eth0.accept_redirects=0
sysctl -w net.ipv4.conf.lo.accept_redirects=0
sysctl -w net.ipv4.conf.default.accept_redirects=0
sysctl -w net.ipv4.conf.all.accept_redirects=0
```

- If this server does not act as a router, it does not need to send redirects, so they can be disabled:

```
sysctl -w net.ipv4.conf.eth0.send_redirects=0
sysctl -w net.ipv4.conf.lo.send_redirects=0
sysctl -w net.ipv4.conf.default.send_redirects=0
sysctl -w net.ipv4.conf.all.send_redirects=0
```

- Configure the server to ignore broadcast pings or smurf attacks:

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

- Ignore all kinds of icmp packets or pings:

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

- Some routers send invalid responses to broadcast frames, and each one generates a warning that is logged by the kernel. These responses can be ignored:

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

Tuning in TCP and UDP

The following commands can be used for tuning servers that support a large number of multiple connections:

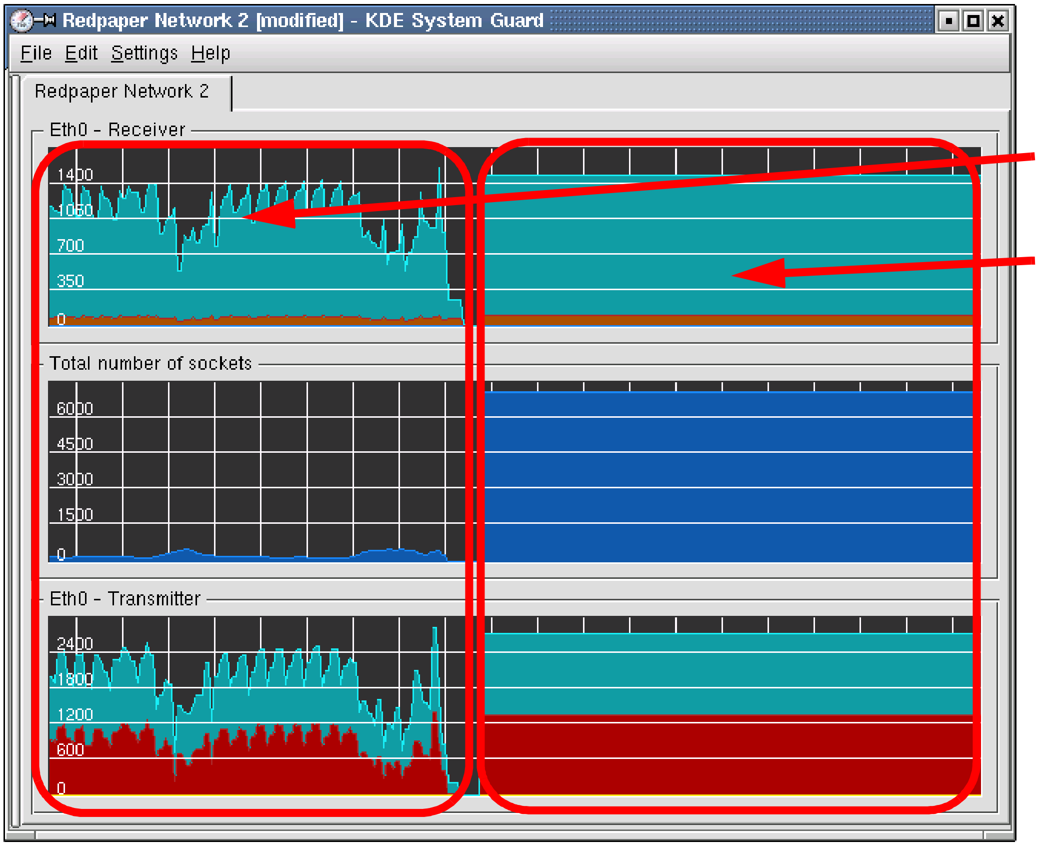
- For servers that receive many connections at the same time, the TIME-WAIT sockets for new connections can be reused. This is useful in Web servers, for example:

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

  If you enable this command, you should also enable fast recycling of TIME-WAIT sockets status:

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

  Figure 1-6 shows that with these parameters enabled, the number of connections is significantly reduced. This is good for performance because each TCP transaction maintains a cache of protocol information about each of the remote clients. In this cache, information such as round-trip time, maximum segment size, and congestion window are stored. For more details, review RFC 1644.



tcp_tw_reuse and **tcp_tw_recycle** enabled.

tcp_tw_reuse and **tcp_tw_recycle** disabled.

With both **tcp_tw_reuse** and **tcp_tw_recycle** enabled, the information about the hosts does not have to be obtained again and the TCP transaction can start immediately, preventing the unnecessary traffic.

Figure 1-6 Parameters reuse and recycle enabled (left) and disabled (right)

- The parameter tcp_fin_timeout is the time to hold a socket in state FIN-WAIT-2 when the socket is closed at the server.

  A TCP connection begins with a three-segment synchronization SYN sequence and ends with a three-segment FIN sequence, neither of which holds data. By changing the tcp_fin_timeout value, the time from the FIN sequence to when the memory can be freed for new connections can be reduced, thereby improving performance. This value, however, should be changed only after careful monitoring, as there is a risk of overflowing memory because of the number of dead sockets.

  ```
  sysctl -w net.ipv4.tcp_fin_timeout=30
  ```

- One of the problems found in servers with many simultaneous TCP connections is the large number of connections that are open but unused. TCP has a keepalive function that probes these connections and, by default, drops them after 7200 seconds (2 hours). This length of time may be too large for your server and may result in excess memory usage and a decrease in server performance.

  Setting it to 1800 seconds (30 minutes), for example, may be more appropriate:

  ```
  sysctl -w net.ipv4.tcp_keepalive_time=1800
  ```

- Set the max OS send buffer size (wmem) and receive buffer size (rmem) to 8 MB for queues on all protocols:

  ```
  sysctl -w net.core.wmem_max=8388608
  sysctl -w net.core.rmem_max=8388608
  ```

  These specify the amount of memory that is allocated for each TCP socket when it is created.

  In addition, you should also use the following commands for send and receive buffers. They specify three values: minimum size, initial size, and maximum size:

  ```
  sysctl -w net.ipv4.tcp_rmem="4096 87380 8388608"
  sysctl -w net.ipv4.tcp_wmem="4096 87380 8388608"
  ```

  The third value must be the same as or less than the value of wmem_max and rmem_max.

- Validate the source packets by reserved path. By default, routers route everything, even packets that obviously are not meant for this network. These packets can be dropped, by enabling the appropriate filter:

  ```
  sysctl -w net.ipv4.conf.eth0.rp_filter=1
  sysctl -w net.ipv4.conf.lo.rp_filter=1
  sysctl -w net.ipv4.conf.default.rp_filter=1
  sysctl -w net.ipv4.conf.all.rp_filter=1
  ```

- When the server is heavily loaded or has many clients with bad connections with high latency, it can result in an increase in half-open connections. This is very common for Web servers, especially when there are many dial-up users. These half-open connections are stored in the *backlog connections* queue. You should set this value to at least 4096 (the default is 1024).

  Setting this value is useful even if your server does not receive this kind of connection, as it can still be protected from a denial-of-service (syn-flood) attack.

  ```
  sysctl -w net.ipv4.tcp_max_syn_backlog=4096
  ```

- We should set the ipfrag parameters particularly for NFS and Samba servers. Here, we can set the maximum and minimum memory used to reassemble IP fragments. When the value of ipfrag_high_thresh in bytes of memory is allocated for this purpose, the fragment handler will drop packets until ipfrag_low_thres is reached.

Fragmentation occurs when there is an error during the transmission of TCP packets. Valid packets are stored in memory (as defined with these parameters) while corrupted packets are retransmitted.

For example, to set the range of available memory to between 256 MB and 384 MB:

```
sysctl -w net.ipv4.ipfrag_low_thresh=262144
sysctl -w net.ipv4.ipfrag_high_thresh=393216
```

2

Tuning tools

Various tools are available with which you can monitor and analyze the performance of your server. Most of these tools use existing information already stored in the /proc directory to present it in a readable format.

In this chapter, we have put together a list of useful command line and graphical tools that are available as packages in SUSE LINUX Enterprise Server or which you can download from the Internet. Three utilities, sar, iostat, and mpstat, are part of the Sysstat package that is provided on SUSE LINUX CD1 or is available from the Sysstat home page:

```
http://perso.wanadoo.fr/sebastien.godard/
```

This chapter includes the following sections:

Table 2-1 lists the functions these tools generally provide.

Table 2-1 Linux performance monitoring tools

Tool	Most useful tool function
uptime	Average system load
dmesg	Hardware/system information
top	Processor activity
iostat	Average CPU load, disk activity
vmstat	System activity
sar	Collect and report system activity
KDE System Guard	Real time systems reporting and graphing
free	Memory usage
traffic-vis	Network monitoring
pmap	Process memory usage
strace	Programs
ulimit	System limits
mpstat	Multiprocessor usage

These tools are in addition to the Capacity Manager tool, which is part of IBM Director. It monitors system performance over a period of time. IBM Director can be used on different operating system platforms and therefore makes it much easier to collect and analyze data in a heterogeneous environment. Capacity Manager is discussed in detail in the redbook *Tuning IBM @server xSeries Servers for Performance*, SG24-5287.

2.1 uptime

The **`uptime`** command can be used to see how long the server has been running and how many users are logged on, as well as for a quick overview of the average load of the server.

The system load average is displayed for the last 1-, 5-, and 15-minute intervals. The load average is not a percentage but the number of processes in queue waiting to be processed. If processes that request CPU time are blocked (which means the CPU has no time to process them), the load average will increase. On the other hand, if each process gets immediate access to CPU time and there are no CPU cycles lost, the load will decrease.

The optimal value of the load is 1, which means that each process has immediate access to the CPU and there are no CPU cycles lost. The typical load can vary from system to system: for a uniprocessor workstation, 1 or 2 might be acceptable, whereas you will probably see values of 8 to 10 on multiprocessor servers.

You can use **`uptime`** to pinpoint a problem with your server or the network. If, for example, a network application is running poorly, run **`uptime`** and you will see whether or not the system load is high. If not, the problem is more likely to be related to your network than to your server.

> **Tip:** You can also use **`w`** instead of **`uptime`**. **`w`** also provides information about who is currently logged on to the machine and what the user is doing.

Example 2-1 Sample output of uptime

```
1:57am  up 4 days 17:05,  2 users,  load average: 0.00, 0.00, 0.00
```

2.2 dmesg

The main purpose of **`dmesg`** is to display kernel messages. **`dmesg`** can provide helpful information in case of hardware problems or problems with loading a module into the kernel.

In addition, with **`dmesg`**, you can determine what hardware is installed in your server. During every boot, Linux checks your hardware and logs information about it. You can view these logs using the command **`/bin/dmesg`**.

Example 2-2 partial output from dmesg

```
EXT3 FS 2.4-0.9.19, 19 August 2002 on sd(8,1), internal journal
EXT3-fs: mounted filesystem with ordered data mode.
IA-32 Microcode Update Driver: v1.11 <tigran@veritas.com>
ip_tables: (C) 2000-2002 Netfilter core team
3c59x: Donald Becker and others. www.scyld.com/network/vortex.html
See Documentation/networking/vortex.txt
01:02.0: 3Com PCI 3c980C Python-T at 0x2080. Vers LK1.1.18-ac
 00:01:02:75:99:60, IRQ 15
  product code 4550 rev 00.14 date 07-23-00
  Internal config register is 3800000, transceivers 0xa.
  8K byte-wide RAM 5:3 Rx:Tx split, autoselect/Autonegotiate interface.
  MII transceiver found at address 24, status 782d.
  Enabling bus-master transmits and whole-frame receives.
01:02.0: scatter/gather enabled. h/w checksums enabled
divert: allocating divert_blk for eth0
ip_tables: (C) 2000-2002 Netfilter core team
```

```
Intel(R) PRO/100 Network Driver - version 2.3.30-k1
Copyright (c) 2003 Intel Corporation

divert: allocating divert_blk for eth1
e100: selftest OK.
e100: eth1: Intel(R) PRO/100 Network Connection
  Hardware receive checksums enabled
  cpu cycle saver enabled

ide-floppy driver 0.99.newide
hda: attached ide-cdrom driver.
hda: ATAPI 48X CD-ROM drive, 120kB Cache, (U)DMA
Uniform CD-ROM driver Revision: 3.12
Attached scsi generic sg4 at scsi1, channel 0, id 8, lun 0,  type 3
```

2.3 top

The **top** command shows you actual processor activity. By default, it displays the most CPU-intensive tasks running on the server and updates the list every five seconds. You can sort the processes by PID (numerically), age (newest first), time (cumulative time) and resident memory usage and time (time the process has occupied the CPU since startup).

Example 2-3 Example output from top command

```
top - 02:06:59 up 4 days, 17:14,  2 users,  load average: 0.00, 0.00, 0.00
Tasks:  62 total,   1 running,  61 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.2% us,  0.3% sy,  0.0% ni, 97.8% id,  1.7% wa,  0.0% hi,  0.0% si
Mem:    515144k total,   317624k used,   197520k free,    66068k buffers
Swap:  1048120k total,       12k used,  1048108k free,   179632k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
13737 root      17   0  1760  896 1540 R  0.7  0.2   0:00.05 top
  238 root       5 -10     0    0    0 S  0.3  0.0   0:01.56 reiserfs/0
    1 root      16   0   588  240  444 S  0.0  0.0   0:05.70 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      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/1
    5 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/1
    6 root       5 -10     0    0    0 S  0.0  0.0   0:00.02 events/0
    7 root       5 -10     0    0    0 S  0.0  0.0   0:00.00 events/1
    8 root       5 -10     0    0    0 S  0.0  0.0   0:00.09 kblockd/0
    9 root       5 -10     0    0    0 S  0.0  0.0   0:00.01 kblockd/1
   10 root      15   0     0    0    0 S  0.0  0.0   0:00.00 kirqd
   13 root       5 -10     0    0    0 S  0.0  0.0   0:00.02 khelper/0
   14 root      16   0     0    0    0 S  0.0  0.0   0:00.45 pdflush
   16 root      15   0     0    0    0 S  0.0  0.0   0:00.61 kswapd0
   17 root      13 -10     0    0    0 S  0.0  0.0   0:00.00 aio/0
   18 root      13 -10     0    0    0 S  0.0  0.0   0:00.00 aio/1
```

You can further modify the processes using **renice** to give a new priority to each process. If a process hangs or occupies too much CPU, you can kill the process (**kill** command).

The columns in the output are as follows:

- PID: Process identification.
- USER: Name of the user who owns (and perhaps started) the process.
- PRI: Priority of the process (see 2.3.1, “Process priority and nice levels” on page 35 for details).
- NI: Niceness level (that is, whether the process tries to be nice by adjusting the priority by the number given; see below for details).
- SIZE: Amount of memory (code+data+stack), in KB, being used by the process.
- RSS: Amount of physical RAM used, in KB.
- SHARE: Amount of memory shared with other processes, in KB.
- STAT: State of the process: S=sleeping, R=running, T=stopped or traced, D=interruptible sleep, Z=zombie. Zombie processes are discussed further in 2.3.2, “Zombie processes” on page 36.
- %CPU: Share of the CPU usage (since the last screen update).
- %MEM: Share of physical memory.
- TIME: Total CPU time used by the process (since it was started).
- COMMAND: Command line used to start the task (including parameters).

Tip: The **`/bin/ps`** command gives you a snapshot view of the current processes.

2.3.1 Process priority and nice levels

Process priority is a number that determines the order in which the process is handled by the CPU. The kernel adjusts this number up and down as needed. The *nice* value is a limit on the priority. The priority number is not allowed to go below the nice value (a lower nice value is a more favored priority).

It is not possible to change the priority of a process. This is only indirectly possible through the use of the nice level of the process. Note that it may not always be possible to change the priority of a process using the nice level. If a process is running too slowly, you can assign more CPU to it by giving it a lower nice level. Of course, this means that all other programs will have fewer processor cycles and will run more slowly.

Linux supports nice levels from 19 (lowest priority) to -20 (highest priority). The default value is 0. To change the nice level of a program to a negative number (which makes it a high priority process), it is necessary to log on or **`su`** to root.

To start the program xyz with a nice level of -5, issue the command:

```
nice -n -5 xyz
```

To change the nice level of a program already running, issue the command:

```
renice level pid
```

If we wanted to change the priority of the xyz program that has a PID of 2500 to a nice level of 10, we would issue the following command:

```
renice 10 2500
```

2.3.2 Zombie processes

When a process has already terminated, having received a signal to do so, it normally takes some time to finish all tasks (such as closing open files) before ending itself. In that normally very short time frame, the process is a *zombie*.

After the process has completed all these shutdown tasks, it reports to the parent process that it is about to terminate. Sometimes, a zombie process is unable to terminate itself, in which case you will see that it has a status of `Z` (zombie).

It is not possible to kill such a process with the **`kill`** command, because it is already considered “dead.” If you cannot get rid of a zombie, you can kill the parent process and then the zombie disappears as well. However, if the parent process is the init process, you should not kill it. The init process is a very important process and therefore a reboot may be needed to get rid of the zombie process.

2.4 iostat

The **`iostat`** command is part of the Sysstat set of utilities, available from:

http://perso.wanadoo.fr/sebastien.godard/

The **`iostat`** command lets you see average CPU times since the system was started, in a way similar to **`uptime`**. In addition, however, **`iostat`** creates a report about the activities of the disk subsystem of the server. The report has two parts: CPU utilization and device (disk) utilization. To use **`iostat`** to perform detailed I/O bottleneck and performance tuning, see 3.4.1, “Finding disk bottlenecks” on page 58.

Example 2-4 Sample output of iostat

```
Linux 2.4.21-9.0.3.EL (x232)    05/11/2004

avg-cpu:  %user   %nice    %sys   %idle
           0.03    0.00    0.02   99.95

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
dev2-0            0.00         0.00         0.04        203       2880
dev8-0            0.45         2.18         2.21     166464     168268
dev8-1            0.00         0.00         0.00         16          0
dev8-2            0.00         0.00         0.00          8          0
dev8-3            0.00         0.00         0.00        344          0
```

The CPU utilization report has four sections:

- %user: Shows the percentage of CPU utilization that was taken up while executing at the user level (applications).
- %nice: Shows the percentage of CPU utilization that was taken up while executing at the user level with a nice priority (priority and nice levels are described in 2.3.1, “Process priority and nice levels” on page 35).
- %sys: Shows the percentage of CPU utilization that was taken up while executing at the system level (kernel).
- %idle: Shows the percentage of time the CPU was idle.

The device utilization report is split into the following sections:

- Device: The name of the block device.

- tps: The number of transfers per second (I/O requests per second) to the device. Multiple single I/O requests can be combined in a transfer request, because a transfer request can have different sizes.
- Blk_read/s, Blk_wrtn/s: Blocks read and written per second indicate data read/written from/to the device in seconds. Blocks may also have different sizes. Typical sizes are 1024, 2048 or 4048 bytes, depending on the partition size. For example, the block size of /dev/sda1 can be found with:

  ```
  dumpe2fs -h /dev/sda1 |grep -F "Block size"
  ```

  This gives an output similar to:

  ```
  dumpe2fs 1.34 (25-Jul-2003)
  Block size:               1024
  ```

- Blk_read, Blk_wrtn: This indicates the total number of blocks read/written since the boot.

2.5 vmstat

vmstat provides information about processes, memory, paging, block I/O, traps and CPU activity.

Example 2-5 Example output from vmstat

```
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 2  0      0 154804  77328 910900    0    0     4     6  103    19  0  0 100  0
```

The columns in the output are as follows:

- Process
  - r: The number of processes waiting for runtime.
  - b: The number of processes in uninterruptable sleep.
  - w: The number of processes swapped out but otherwise runable. This field is calculated.
- Memory
  - swpd: The amount of virtual memory used (KB).
  - free: The amount of idle memory (KB).
  - buff: The amount of memory used as buffers (KB).
- Swap
  - si: Amount of memory swapped from the disk (KBps).
  - so: Amount of memory swapped to the disk (KBps).
- IO
  - bi: Blocks sent to a block device (blocks/s).
  - bo: Blocks received from a block device (blocks/s).
- System
  - in: The number of interrupts per second, including the clock.
  - cs: The number of context switches per second.
- CPU (these are percentages of total CPU time)
  - us: Time spent running non-kernel code (user time, including nice time).
  - sy: Time spent running kernel code (system time).

- id: Time spent idle. Prior to Linux 2.5.41, this included IO-wait time.
- Time spent waiting for IO. Prior to Linux 2.5.41, this appeared as zero.

2.6 sar

The **sar** command is part of the Sysstat set of utilities, available from:

http://perso.wanadoo.fr/sebastien.godard/

The **sar** command is used to collect, report or save system activity information. The **sar** command consists of three applications: **sar** which displays the data, and **sa1** and **sa2** which are used for collecting and storing the data.

By using **sa1** and **sa2**, the system can be configured to get the information and log it for later analysis. To do this, a cron job must be configured. To accomplish this, add the following lines to the /etc/crontab (see Example 2-6).

Example 2-6 Example of starting automatic log reporting with cron

```
# 8am-7pm activity reports every 10 minutes during weekdays.
*/10 8-18 * * 1-5 /usr/lib/sa/sa1 600 6 &
# 7pm-8am activity reports every an hour during weekdays.
0 19-7 * * 1-5 /usr/lib/sa/sa1 &
# Activity reports every an hour on Saturday and Sunday.
0 * * * 0,6 /usr/lib/sa/sa1 &
# Daily summary prepared at 19:05
5 19 * * * /usr/lib/sa/sa2 -A &
```

Alternatively, you can use **sar** to run almost real-time reporting from the command line, as shown in Example 2-7.

From the collected data, you get a detailed overview of your CPU utilization (%user, %nice, %system, %idle), memory paging, network I/O and transfer statistics, process creation activity, activity for block devices, and interrupts/second over time.

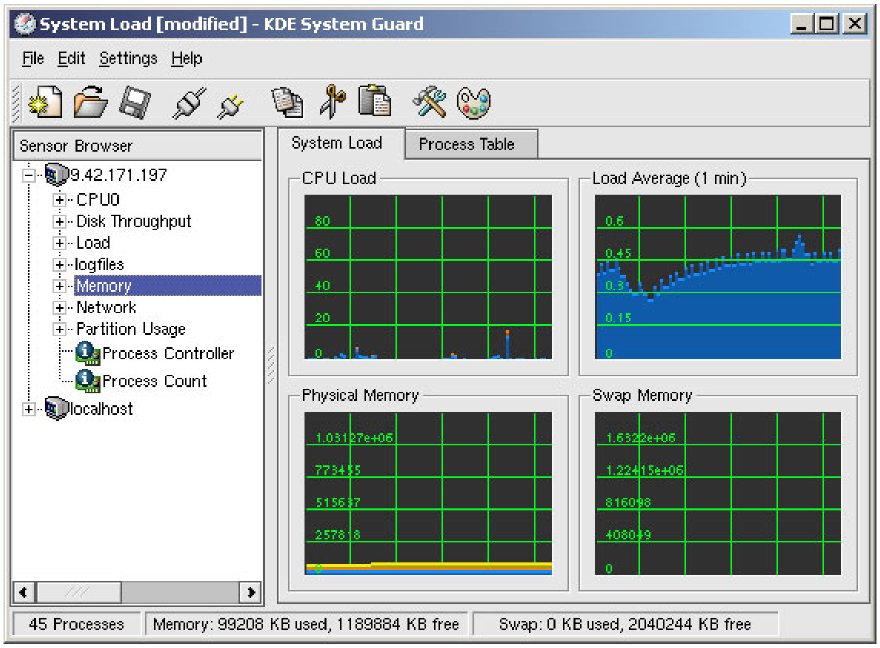
Example 2-7 Ad hoc CPU monitoring

```
[root@x232 root]# sar -u 3 10
Linux 2.4.21-9.0.3.EL (x232)    05/22/2004

02:10:40 PM       CPU     %user     %nice   %system     %idle
02:10:43 PM       all      0.00      0.00      0.00    100.00
02:10:46 PM       all      0.33      0.00      0.00     99.67
02:10:49 PM       all      0.00      0.00      0.00    100.00
02:10:52 PM       all      7.14      0.00     18.57     74.29
02:10:55 PM       all     71.43      0.00     28.57      0.00
02:10:58 PM       all      0.00      0.00    100.00      0.00
02:11:01 PM       all      0.00      0.00      0.00      0.00
02:11:04 PM       all      0.00      0.00    100.00      0.00
02:11:07 PM       all     50.00      0.00     50.00      0.00
02:11:10 PM       all      0.00      0.00    100.00      0.00
Average:          all      1.62      0.00      3.33     95.06
```

2.7 KDE System Guard

KDE System Guard (KSysguard) is the KDE task manager and performance monitor. It features a client/server architecture that enables monitoring of local as well as remote hosts.



45 Processes | Memory: 99208 KB used, 1189884 KB free | Swap: 0 KB used, 2040244 KB free

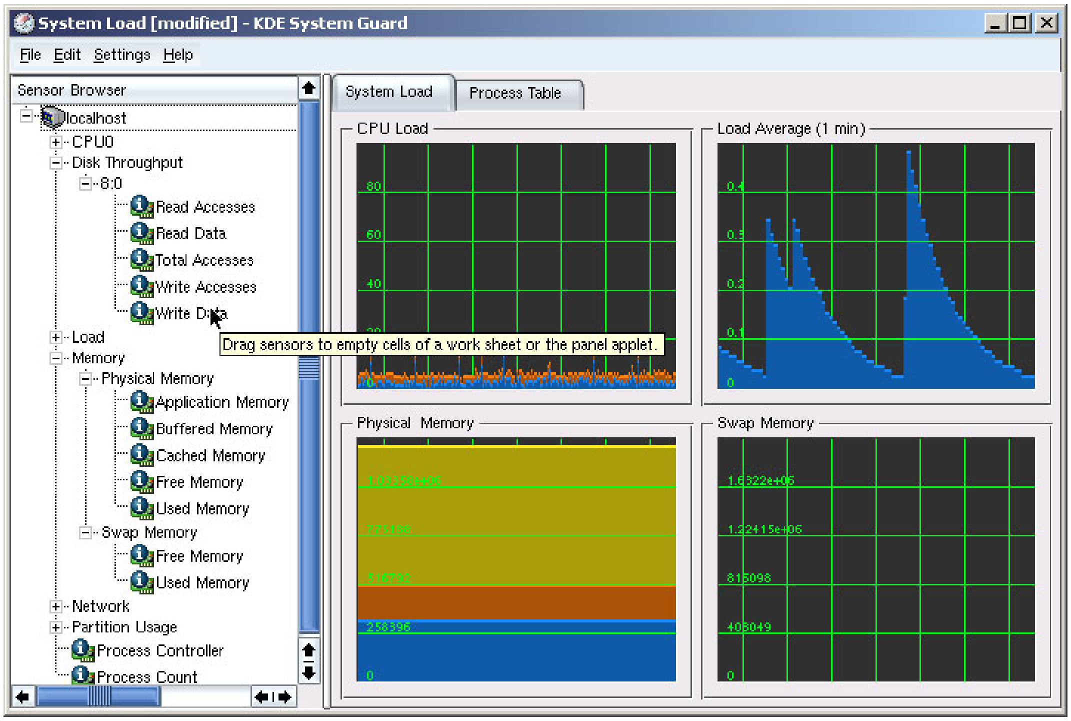
Figure 2-1 Default KDE System Guard window

The graphical front end (Figure 2-1) uses *sensors* to retrieve the information it displays. A sensor can return simple values or more complex information such as tables. For each type of information, one or more displays are provided. Displays are organized in worksheets that can be saved and loaded independently of each other.

The KSysguard main window (Figure 2-1) consists of a menu bar, an optional tool bar and status bar, the sensor browser, and the work space. When first started, you will see your local machine listed as localhost in the sensor browser and two tabs in the work space area. This is the default setup.

Each sensor monitors a certain system value. All of the displayed sensors can be dragged and dropped in the work space. There are three options:

- You can delete and replace sensors in the actual work space.
- You can edit work sheet properties and increase the number of row and or columns.
- You can create a new worksheet and drop new sensors meeting your needs.



Drag sensors to empty cells of a work sheet or the panel applet.

Figure 2-2 KDE System Guard Sensor Browser

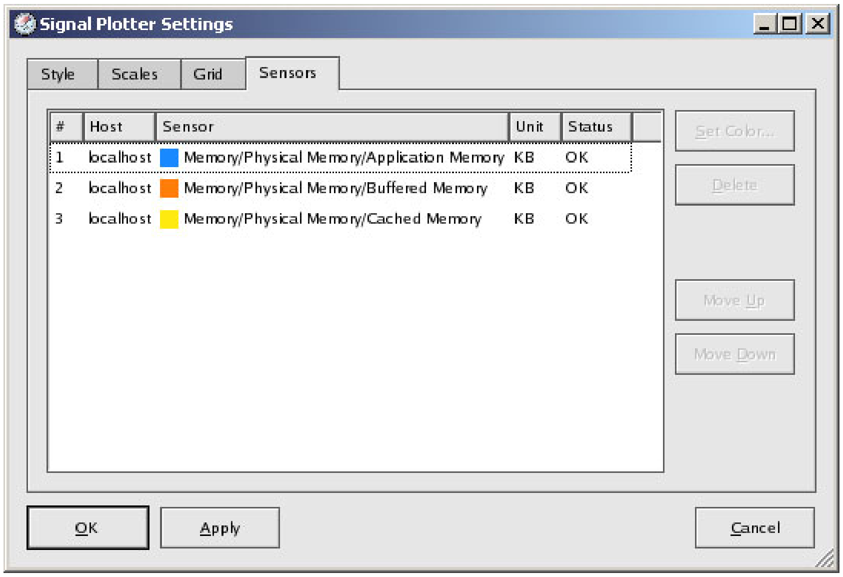
2.7.1 Work space

By looking at the work space in Figure 2-2, you notice that there are two tabs:

- System Load, the default view when first starting up KSysguard
- Process Table

System Load

The System Load worksheet consists of four sensor windows; these are CPU Load, Load Average (1 Minute), Physical Memory, and Swap Memory. You will note from the Physical Memory window that it is possible to have multiple sensors displayed within one window. To determine which sensors are being monitored in a given window, mouse over the graph and some descriptive text will appear. Another way to do this is to right-click the graph and click **Properties**, and then click the **Sensors** tab, as shown in Figure 2-3 on page 41. This also shows a key of what each color represents on the graph.

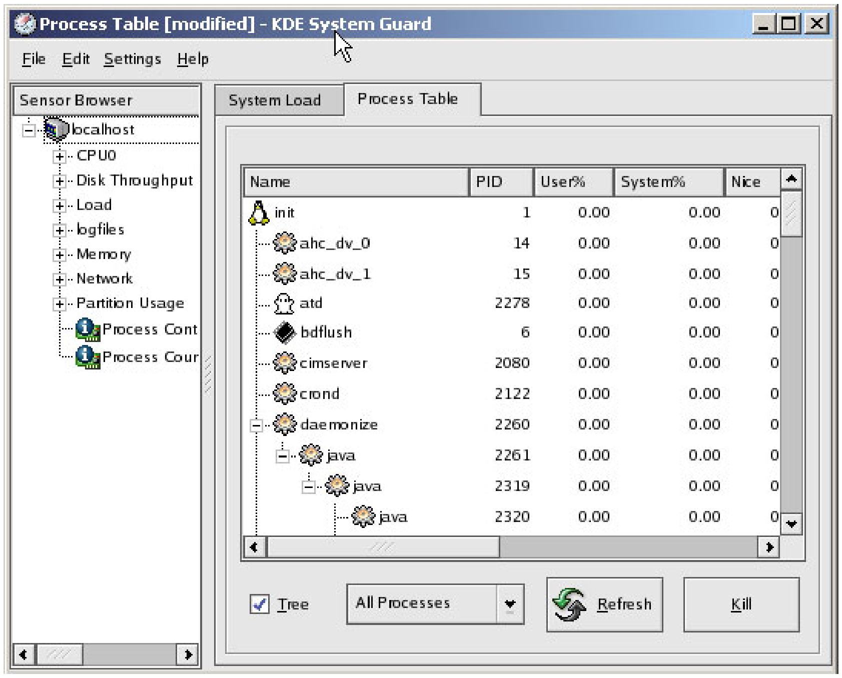


#	Host	Sensor	Unit	Status
1	localhost	🟦 Memory/Physical Memory/Application Memory	KB	OK
2	localhost	🟧 Memory/Physical Memory/Buffered Memory	KB	OK
3	localhost	🟨 Memory/Physical Memory/Cached Memory	KB	OK

Figure 2-3 Sensor Information, Physical Memory Signal Plotter

Process Table

Clicking the **Process Table** tab displays information about all the running processes on the server (Figure 2-4). The table, by default, is sorted by System CPU utilization, but this can be changed by simply clicking the heading by which you wish to sort.



Name	PID	User%	System%	Nice
init	1	0.00	0.00	0
ahc_dv_0	14	0.00	0.00	0
ahc_dv_1	15	0.00	0.00	0
atd	2278	0.00	0.00	0
bdflush	6	0.00	0.00	0
cimserver	2080	0.00	0.00	0
crond	2122	0.00	0.00	0
daemonize	2260	0.00	0.00	0
java	2261	0.00	0.00	0
java	2319	0.00	0.00	0
java	2320	0.00	0.00	0

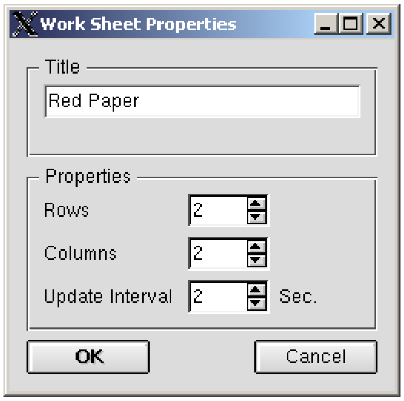
Figure 2-4 Process Table view

Configuring a work sheet

For your environment or the particular area that you wish to monitor, it may be necessary to use different sensors for monitoring. The best way to do this is to create a custom work sheet. In this section, we guide you through the steps that are required to create the work sheet shown in Figure 2-7 on page 43.

The steps to create a worksheet are as follows:

1. Create a blank worksheet by clicking **File** → **New**; this opens the window shown in Figure 2-5.



Title
Red Paper

Properties
Rows 2
Columns 2
Update Interval 2 Sec.

Figure 2-5 Properties for new worksheet

2. Enter a title and a number of rows and columns; this gives you the maximum number of monitor windows, which in our case will be four. When the information is complete, click **OK** to create the blank worksheet, as shown in Figure 2-6.

> **Note:** The fastest update interval that can be defined is two seconds.

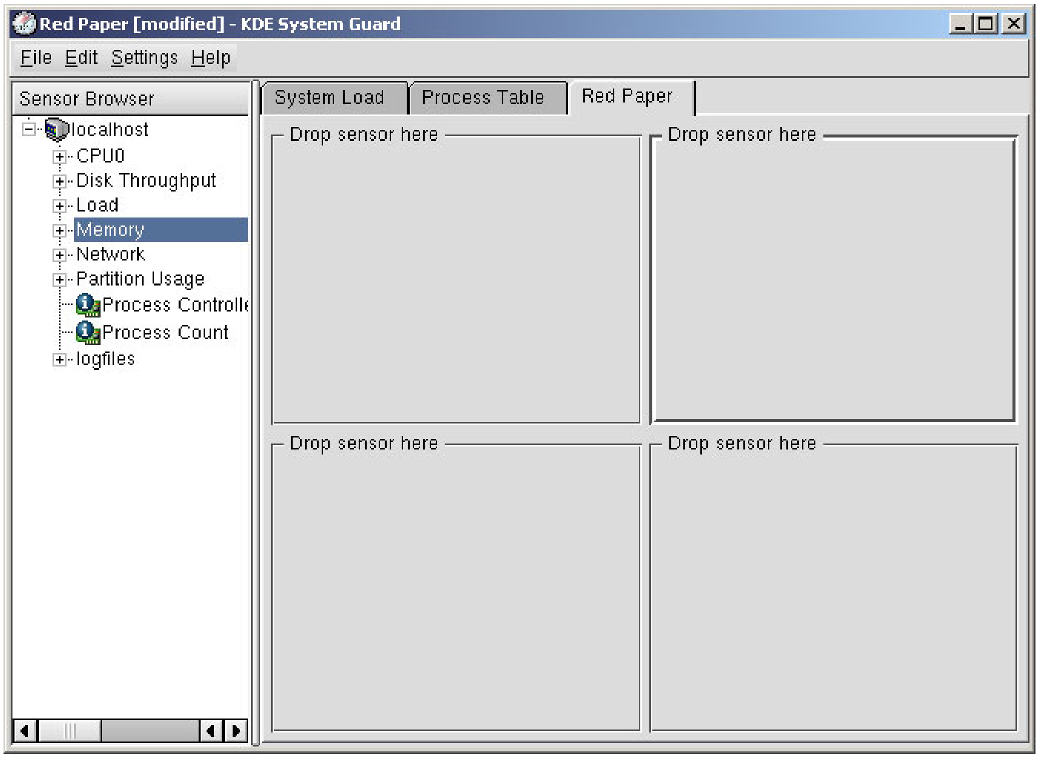


Figure 2-6 Empty worksheet

3. Now you can fill in the sensor boxes by simply dragging the sensors on the left side of the window to the desired box on the right.

   The display choices are:

   – Signal Plotter: This sensor style displays samples of one or more sensors over time. If several sensors are displayed, the values are layered in different colors. If the display is large enough, a grid will be displayed to show the range of the plotted samples.

     By default, the automatic range mode is active, so the minimum and maximum values will be set automatically. If you want fixed minimum and maximum values, you can deactivate the automatic range mode and set the values in the Scales tab from the Properties dialog window (which you access by right-clicking the graph).

   – Multimeter: The Multimeter displays the sensor values as a digital meter. In the properties dialog, you can specify a lower and upper limit. If the range is exceeded, the display is colored in the alarm color.

   – BarGraph: The BarGraph displays the sensor value as dancing bars. In the properties dialog, you can also specify the minimum and maximum values of the range and a lower and upper limit. If the range is exceeded, the display is colored in the alarm color.

   – Sensor Logger: The Sensor Logger does not display any values, but logs them in a file with additional date and time information.

     For each sensor, you have to define a target log file, the time interval the sensor will be logged and whether alarms are enabled.

4. Click **File** → **Save** to save the changes to the worksheet.

> **Note:** When you save a work sheet, it will be saved in the user's home directory, which may prevent other administrators from using your custom worksheets.

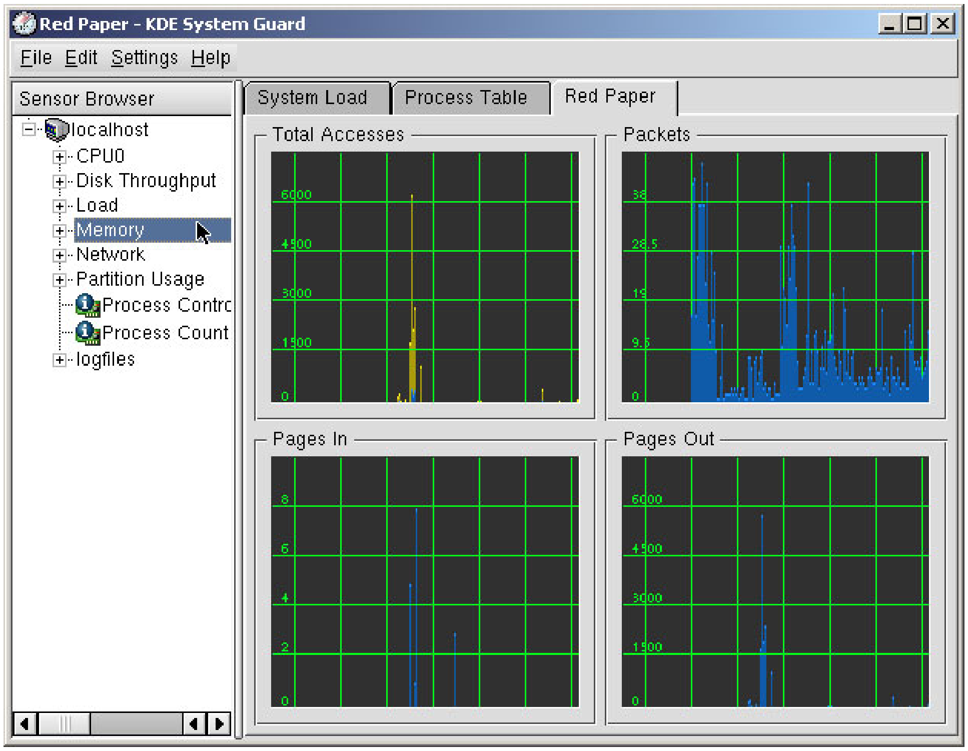


Figure 2-7 Example worksheet

More information about KDE System Guard can be found online at:

http://docs.kde.org/en/3.2/kdebase/ksysgaurd

2.8 free

The command **/bin/free** displays information about the total amounts of free and used memory (including swap) on the system. It also includes information about the buffers and cache used by the kernel.

Example 2-8 Example output from the free command

```
             total       used       free     shared    buffers     cached
Mem:       1291980     998940     293040          0      89356     772016
-/+ buffers/cache:     137568    1154412
Swap:      2040244          0    2040244
```

2.9 Traffic-vis

Traffic-vis is a suite of tools which determine which hosts have been communicating on an IP network, with whom they have been communicating, and the volume of communication that has taken place. The final report can be generated in plain text, HTML or GIF.

Start the program to collect data on interface eth0, for example:

```
traffic-collector -i eth0 -s /root/output_traffic-collector
```

After the program starts, it is detached from the terminal and begins the collection of data. You can control the program by using the **killall** command to send a signal to the process. For example, to write the report to disk, issue the command:

```
killall -SIGUSR1 traffic-collector
```

To stop the collection of data:

```
killall -SIGTERM traffic-collector
```

> **Tip:** Do not forget to run this last command; otherwise, your system’s performance will degrade due to a lack of memory.

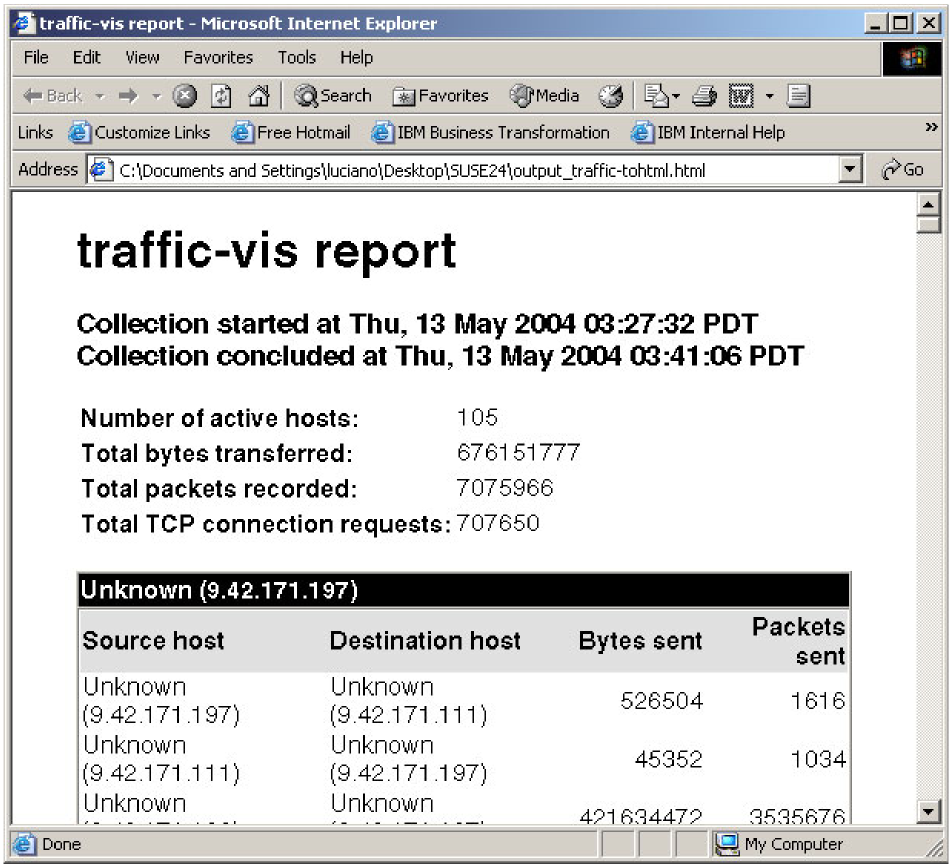
You can sort the output by packets, bytes, TCP connections, the total of each one, or the number of sent and/or received of each one. For example, to sort total packets sent and received on hosts:

```
traffic-sort -i output_traffic-collector -o output_traffic-sort -Hp
```

To generate a report in HTML format which displays the total bytes transferred, total packets recorded, total TCP connections requests, and other information about each server on the network, run:

```
traffic-tohtml -i output_traffic-sort -o output_traffic-tohtml.html
```

This output file can be displayed in a browser, as shown in Figure 2-8.


traffic-vis report - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Links Customize Links Free Hotmail IBM Business Transformation IBM Internal Help
Address C:\Documents and Settings\luciano\Desktop\SUSE24\output_traffic-tohtml.html
Go
Done
My Computer

traffic-vis report

Collection started at Thu, 13 May 2004 03:27:32 PDT
Collection concluded at Thu, 13 May 2004 03:41:06 PDT

Number of active hosts:	105
Total bytes transferred:	676151777
Total packets recorded:	7075966
Total TCP connection requests:	707650

Unknown (9.42.171.197)

Source host	Destination host	Bytes sent	Packets sent
Unknown (9.42.171.197)	Unknown (9.42.171.111)	526504	1616
Unknown (9.42.171.111)	Unknown (9.42.171.197)	45352	1034
Unknown	Unknown	421634472	3535676

Figure 2-8 Report generated by traffic-vis

To generate a report in GIF format with a width of 600 pixels and a height of 600 pixels, use the following command:

```
traffic-togif -i output_traffic-sort -o output_traffic-togif.gif -x 600 -y 600
```

This graphic (Figure 2-9 on page 46) shows the communication between systems on the network. You can also see that some hosts talk to others, but there are servers that never talk to each other.

This output is typically used to find broadcasts on the network. To see what servers are using IPX/SPX protocol in a TCP network and separate both networks, remember that IPX is based on broadcast packets. If we need to pinpoint others types of problems, such as damaged network cards or duplicated IPs on networks, we need to use more specific tools such as Ethereal, which is installed by default on SUSE LINUX Enterprise Server.

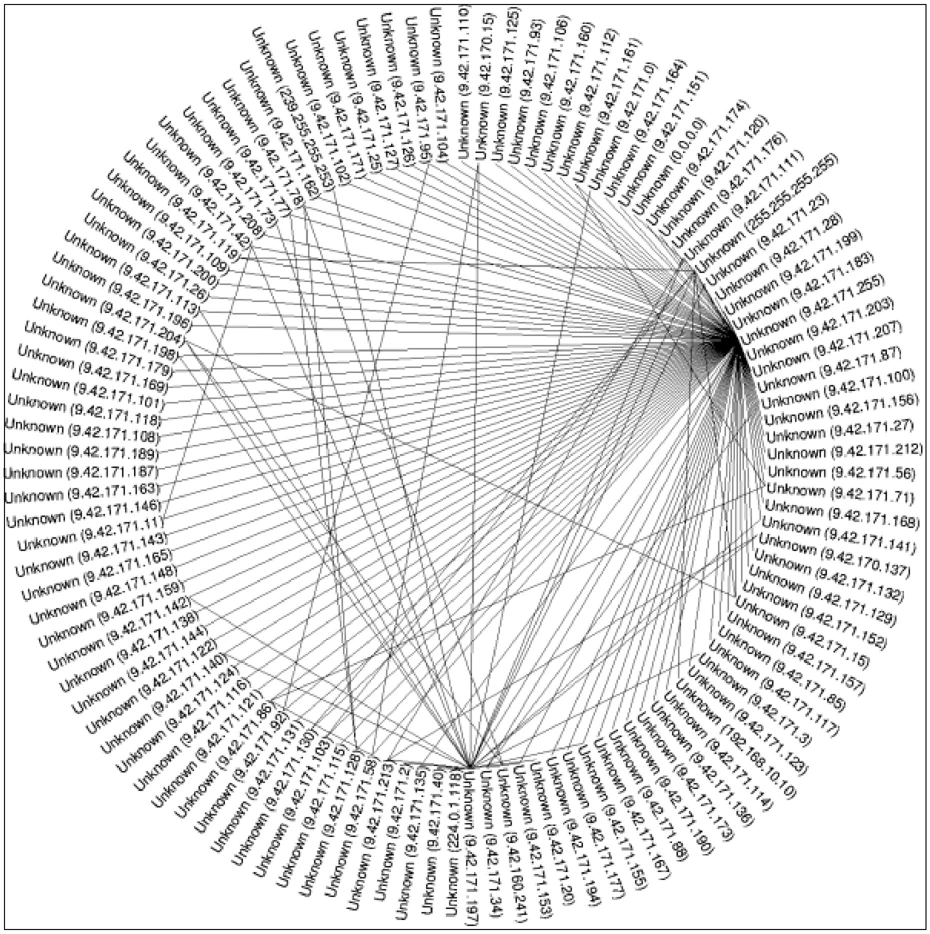


Figure 2-9 Report generate by traffic-vis

Tip: Using pipes, it is possible to produce output in one command. For example, to generate a report in HTML, run:

```
cat output_traffic-collector | traffic-sort -Hp | traffic-tohtml -o
output_traffic-tohtml.html
```

To generate a report as a GIF file, run:

```
cat output_traffic-collector | traffic-sort -Hp | traffic-togif -o
output_traffic-togif.gif -x 600 -y 600
```

2.10 pmap

The **pmap** command reports the amount of memory that one or more processes are using. You can use this tool to determine which processes on the server are being allocated memory and whether this amount of memory is a cause of memory bottlenecks.

```
pmap -x <pid>
```

Example 2-9 Total amount of memory cupsd process is using

```
linux:~ # pmap -x 1796
1796:   /usr/sbin/cupsd
Address   Kbytes     RSS    Anon  Locked Mode   Mapping
08048000     244       -       -       - r-x--  cupsd
ffffe000       4       -       -       - -----    [ anon ]
-------- ------- ------- ------- -------
total kB    6364       -       -       -
```

For the complete syntax of the **pmap** command, issue:

```
pmap -?
```

2.11 strace

The **strace** command intercepts and records the system calls that are called by a process, and the signals that are received by a process. This is a useful diagnostic, instructional, and debugging tool. System administrators will find it valuable for solving problems with programs. For more information, see Chapter 4, “Tuning Apache” on page 65.

To use the command, specify the process ID (PID) to be monitored:

```
strace -p <pid>
```

Example 2-10 on page 48 shows an example of the output of **strace**.

Example 2-10 Output of strace monitoring httpd process, for example

```
[root@x232 html]# strace -p 815
Process 815 attached - interrupt to quit
semop(360449, 0xb73146b8, 1)            = 0
poll([{fd=4, events=POLLIN}, {fd=3, events=POLLIN, revents=POLLIN}], 2, -1) = 1
accept(3, {sa_family=AF_INET, sin_port=htons(52534), sin_addr=inet_addr("9.42.171.197")}, [16]) = 13
semop(360449, 0xb73146be, 1)            = 0
getsockname(13, {sa_family=AF_INET, sin_port=htons(80), sin_addr=inet_addr("9.42.171.198")}, [16]) = 0
fcntl64(13, F_GETFL)                    = 0x2 (flags O_RDWR)
fcntl64(13, F_SETFL, O_RDWR|O_NONBLOCK) = 0
read(13, 0x8259bc8, 8000)               = -1 EAGAIN (Resource temporarily unavailable)
poll([{fd=13, events=POLLIN, revents=POLLIN}], 1, 300000) = 1
read(13, "GET /index.html HTTP/1.0\r\nUser-A"..., 8000) = 91
gettimeofday({1084564126, 750439}, NULL) = 0
stat64("/var/www/html/index.html", {st_mode=S_IFREG|0644, st_size=152, ...}) = 0
open("/var/www/html/index.html", O_RDONLY) = 14
mmap2(NULL, 152, PROT_READ, MAP_SHARED, 14, 0) = 0xb7052000
writev(13, [{"HTTP/1.1 200 OK\r\nDate: Fri, 14 M"..., 264}, {"<html>\n<title>\n     RedPaper Per"..., 152}], 2) = 416
munmap(0xb7052000, 152)                 = 0
socket(PF_UNIX, SOCK_STREAM, 0)         = 15
connect(15, {sa_family=AF_UNIX, path="/var/run/.nscd_socket"}, 110) = -1 ENOENT (No such file or directory)
close(15)                               = 0
```

For the complete syntax of the **strace** command, issue:

```
strace -?
```

2.12 ulimit

This command is built into the bash shell and is used to provide control over the resources available to the shell and to the processes started by it on systems that allow such control.

Use the option **-a** to list all parameters that we can set:

```
ulimit -a
```

Example 2-11 Output of ulimit

```
linux:~ # ulimit -a
core file size        (blocks, -c) 0
data seg size         (kbytes, -d) unlimited
file size             (blocks, -f) unlimited
max locked memory     (kbytes, -l) unlimited
max memory size       (kbytes, -m) unlimited
open files                    (-n) 1024
pipe size          (512 bytes, -p) 8
stack size            (kbytes, -s) unlimited
cpu time             (seconds, -t) unlimited
max user processes            (-u) 4095
virtual memory        (kbytes, -v) unlimited
```

The **-H** and **-S** options specify the hard and soft limits that can be set for the given resource. If the soft limit is passed, the system administrator will receive a warning. The hard limit is the maximum value which can be reached before the user gets the error messages `Out of file handles`.

For example, you can set a hard limit for the number of file handles and open files (-n):

```
ulimit -Hn 4096
```

For the soft limit of number of file handles and open files, use:

```
ulimit -Sn 1024
```

To see the hard and soft values, issue the command with a new value:

```
ulimit -Hn
ulimit -Sn
```

This command can be used, for example, to limit Oracle users on the fly. To set it on startup, enter the follow lines, for example, in /etc/security/limits.conf:

```
soft nofile 4096
hard nofile 10240
```

In addition, make sure that the files /etc/pam.d/login and /etc/pam.d/sshd have the following entry:

```
session  required       pam_limits.so
```

This entry is required so that the system can enforce these limits.

For the complete syntax of the **ulimit** command, issue:

```
ulimit -?
```

2.13 mpstat

The **mpstat** command is part of the Sysstat set of utilities, available from:

http://perso.wanadoo.fr/sebastien.godard/

The **mpstat** command is used to report the activities of each the CPUs available on a multiprocessor server. Global average activities among all CPUs are also reported.

For example, use the follow command to display three entries of global statistics among all processors at two second intervals:

```
mpstat 2 3
```

Tip: This can be used in non-SMP machines as well.

Example 2-12 Output of mpstat command on uni-processor machine (xSeries 342)

```
x342rsa:~ # mpstat 2 3
Linux 2.4.21-215-default (x342rsa)      05/20/04

07:12:16     CPU   %user   %nice %system   %idle    intr/s
07:12:34     all    1.00    0.00    1.50   97.50    104.00
07:12:36     all    1.00    0.00    1.50   97.50    104.50
07:12:38     all    1.00    0.00    1.50   97.50    104.00
Average:     all    1.10    0.00    1.55   97.35    103.80
```

To display three entries of statistics for all processors of a multi-processor server at one second intervals, use the command:

```
mpstat -P ALL 1 3
```

Example 2-13 Output of mpstat command on four-way machine (xSeries 232)

```
[root@x232 root]# mpstat -P ALL 1 10
Linux 2.4.21-9.0.3.EL (x232)    05/20/2004

02:10:49 PM  CPU   %user   %nice %system   %idle    intr/s
02:10:50 PM  all    0.00    0.00    0.00  100.00    102.00
02:10:51 PM  all    0.00    0.00    0.00  100.00    102.00
02:10:52 PM    0    0.00    0.00    0.00  100.00    102.00
Average:     all    0.00    0.00    0.00  100.00    103.70
Average:       0    0.00    0.00    0.00  100.00    103.70
```

For the complete syntax of the **mpstat** command, issue:

```
mpstat -?
```

3

Analyzing performance bottlenecks

This chapter is useful if you are facing a situation where a performance problem is already affecting a server. This is a reactive situation in which you need to follow a series of steps to lead you to a concrete solution that you can implement to restore the server to an acceptable performance level.

The topics that are covered in this chapter are:

3.1 Identifying bottlenecks

The following steps are used as our quick tuning strategy:

1. Know your system
2. Back up the system
3. Monitor and analyze the system performance
4. Narrow down the bottleneck and find its cause
5. Fix the bottleneck cause by trying only one single change at a time
6. Go back to step 3 until you are satisfied with the performance of the system

> **Tip:** You should document each step, especially the changes you make and their effect on performance.

3.1.1 Gathering information

Mostly likely, the only first-hand information you will have access to will be comprised of statements such as “There is a problem with the server.” It is crucial to use probing questions to clarify and document the problem. Here is a list of questions you should ask to help you get a better picture of the system.

- Can you give me a complete description of the server in question?
  - Model
  - Age
  - Configuration
  - Peripheral equipment
  - Operating system version and update level
- Can you tell me what the problem is *exactly*?
  - What are the symptoms?
  - Description of any error messages.

  Some people will have problems answering this question. Any extra information the customer can give you might enable you to find the problem. For example, the customer might say “It is really slow when I copy large files to the server.” This might indicate a network problem or a disk subsystem problem.
- Who is experiencing the problem?

  Is one person, one particular group of people, or the entire organization experiencing the problem? This will help you determine whether the problem exists in one particular part of the network, whether it is application-dependent, and so forth. If only one user is experiencing the problem, the problem might be with the user’s PC (or their imagination).

  The perception clients have of the server is usually a key factor. From this point of view, performance problems may not be directly related to the server: the network path between the server and the clients can easily be the cause of the problem. This path includes network devices as well as services provided by other servers, such as domain controllers.
- Can the problem be reproduced?

  All reproducible problems can be solved. If you have sufficient knowledge of the system, you should be able to narrow the problem to its root and decide which actions should be taken.

The fact that the problem can be reproduced will enable you to see and understand it better. Document the sequence of actions necessary to reproduce the problem at any time:

  - What are the steps to reproduce the problem?

    Knowing the steps may let you reproduce the same problem on a different machine under the same conditions. If this works, it gives you the opportunity to use a machine in a test environment and removes the chance of crashing the production server.

  - Is it an intermittent problem?

    If the problem is intermittent, the first thing to do is to gather information and find a path to move the problem in the reproducible category. The goal here is to have a scenario to make the problem happen on command.

  - Does it occur at certain times of the day or certain days of the week?

    This might help you determine what is causing the problem. It may occur when everyone arrives for work or returns from lunch. Look for ways to change the timing (that is, make it happen less or more often); if there are ways to do so, the problem becomes a reproducible one.

  - Is it unusual?

    If the problem falls into the non-reproducible category, you may conclude that it is the result of extraordinary conditions and classify it as fixed. In real life, there is a high probability that it will happen again.

    A good procedure to troubleshoot a hard-to-reproduce problem is to perform general maintenance on the server: reboot, or bring the machine up to date on drivers and patches.

- When did the problem start? Was it gradual or did it occur very quickly?

  If the performance issue appeared gradually, it is likely to be a sizing issue; if it appeared overnight, the problem could be caused by a change made to the server or peripherals.

- Have any changes been made to the server (minor or major) or are there any changes in the way clients are using the server?

  Did the customer alter something on the server or peripherals to cause the problem? Is there a log of all network changes available?

Demands could change based on business changes, which could affect demands on a servers and network systems.

- Are there any other servers or hardware components involved?
- Are there any logs available?
- What is the priority of the problem? When does it need to be fixed?
  - Does it need to be fixed in the next few minutes, or in days? You may have some time to fix it; or it may already be time to operate in panic mode.
  - How massive is the problem?
  - What is the related cost of that problem?

3.1.2 Analyzing the server's performance

> **Important:** Before taking any troubleshooting actions, back up all data and the configuration information to prevent a partial or complete lost.

At this point, you should begin monitoring the server. The simplest way is to run monitoring tools from the server that is being analyzed. See Chapter 2, "Tuning tools" on page 31, for information about monitoring tools.

A performance log of the server should be created during its peak time of operation (for example, 9:00 a.m. to 5:00 p.m.); it will depend on what services are being provided and on who is using these services. When creating the log, if available, the following objects should be included:

- Processor
- System
- Server work queues
- Memory
- Page file
- Physical disk
- Redirector
- Network interface

Before you begin, remember that a methodical approach to performance tuning is important. Our recommended process, which you can use for your xSeries server performance tuning process, is as follows:

1. Understand the factors affecting server performance. This Redpaper and the redbook *Tuning IBM @server xSeries Servers for Performance*, SG24-5287 can help.
2. Measure the current performance to create a performance baseline to compare with your future measurements and to identify system bottlenecks.
3. Use the monitoring tools to identify a performance bottleneck. By following the instructions in the next sections, you should be able to narrow down the bottleneck to the subsystem level.
4. Improve the component causing the bottleneck by performing some actions to improve server performance in response to demands.

   > **Note:** It is important to understand that the greatest gains are obtained by upgrading a component which has a bottleneck when the other components in the server have ample "power" left to sustain an elevated level of performance.

5. Measure the new performance. This helps you compare the performance before and after the tuning steps.

When attempting to fix a performance problem, remember the following:

- Take measurements before you upgrade or modify anything so that you can tell whether the change had any effect (that is, take baseline measurements).
- Examine the options that involve reconfiguring existing hardware, not just those that involve adding new hardware.

3.2 CPU bottlenecks

For servers whose primary role is that of an application or database server, the CPU is a critical resource and can often be a source of performance bottlenecks. It is important to note that high CPU utilization does not always mean that a CPU is busy doing work; it may, in fact, be waiting on another subsystem. When performing proper analysis, it is very important that you look at the system as a whole and look at all subsystems because there may be a cascade effect within the subsystems.

Note: There is, however, a common misconception that the CPU is the most important part of the server. Unfortunately, this is often not the case and, as such, servers are often overconfigured with CPU and underconfigured with disks, memory, and network subsystems. Only specific applications that are truly CPU-intensive can take advantage of today's high-end processors.

3.2.1 Finding CPU bottlenecks

Determining bottlenecks with the CPU can be accomplished in several ways. As we have discussed in Chapter 2, "Tuning tools" on page 31, Linux has a variety of tools to help determine this; the question is, which tools to use?

One such tool is **`uptime`**. By analyzing the output from **`uptime`**, we can get a rough idea of what has been happening in the system for the last 15 minutes. For a more detailed explanation of this tool, see 2.1, "uptime" on page 33.

Example 3-1 uptime output from a CPU strapped system

```
18:03:16  up 1 day,  2:46,  6 users,  load average: 182.53, 92.02, 37.95
```

Using KDE System Guard and the CPU sensors lets you view the current CPU workload.

Tip: Be careful not to add to CPU problems by running too many tools at one time. You may find that using a lot of different monitoring tools at one time may be contributing to the high CPU load.

Using **top**, you can not only see CPU utilization but also what processes are the biggest contributor to the problem, as shown in Example 2-3 on page 34. If you have set up **`sar`**, you are collecting a lot of information, some of which is CPU utilization over a period of time. Analyzing this information can be difficult, so use **`isag`**, which can take **`sar`** output and plot a graph. Otherwise, you may wish to parse the information through a script and use a spreadsheet to plot it to see any trends in CPU utilization. You can also use **`sar`** from the command line by issuing **sar -u** or **sar -U** *processornumber*. To gain a broader perspective of the system and current utilization of more than just the CPU subsystem, a good tool is **`vmstat`**, which is described in greater detail in 2.5, "vmstat" on page 37.

3.2.2 SMP

SMP-based systems can present their own set of interesting problems that can be hard to detect. In an SMP environment, there is the concept of CPU affinity. *CPU affinity* implies that you bind a process to a CPU.

The main reason this is useful is CPU cache optimization, which is achieved by keeping the same process on one CPU rather than moving between processors. When a process moves between CPUs, the cache of the new CPU must be flushed, so a process that moves between processors causes many cache flushes to occur, which means that an individual process will take longer to finish. This scenario is very hard to detect because it will appear, when monitoring it, that the CPU load is very balanced and not necessarily peaking on any CPU. Affinity is also useful in NUMA-based systems such as the xSeries 445 and xSeries 455, where it is important to keep memory, cache, and CPU access local to one another.

3.2.3 Performance tuning options

The first step is to ensure that the system performance problem is being caused by the CPU and not one of the other subsystems. If it is the processor that is the server bottleneck then a number of steps can be taken to improve performance. These include the following:

- Ensure that no unnecessary programs are running in the background by using **`ps -ef`**. If you find such programs, stop them and use **`cron`** to schedule them to run at off-peak hours.
- Identify non-critical, CPU-intensive processes by using **`top`** and modify their priority using **`renice`**.
- In an SMP-based machine, try using **`taskset`** to bind processes to CPUs to make sure that processes are not hopping between processors, causing cache flushes.
- Based on the application running, it may be better to scale up (bigger CPUs) than scale out (more CPUs). This is a function of whether your application was designed to effectively take advantage of more processors. For example, a single-threaded application would scale better with a faster CPU and not with more CPUs.
- General options include making sure you are using the latest drivers and firmware, because this might affect the load they have on the CPU.

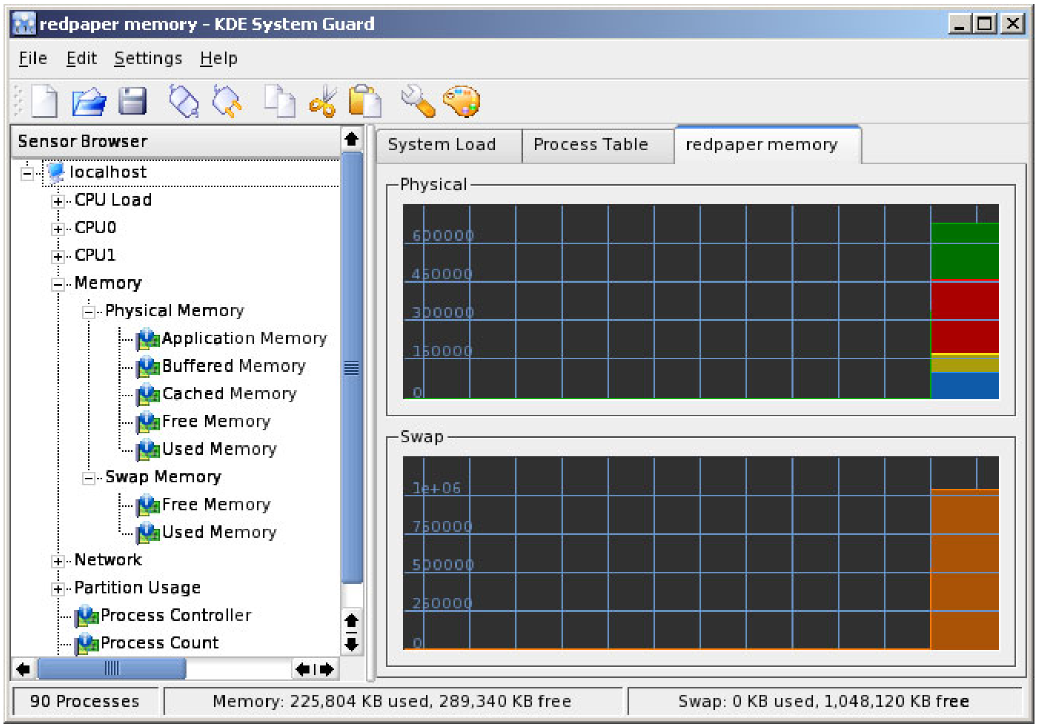
3.3 Memory bottlenecks

On a Linux system, many programs run at the same time; these programs support multiple users and some processes are more used than others. Some of these programs use a portion of memory while the rest are “sleeping.” When an application accesses cache, the performance increases because an in-memory access retrieves data, thereby eliminating the need to access slower disks.

The OS uses an algorithm to control what programs will use physical memory, and which are paged out. This is transparent to user programs. Page space is a file created by the OS on a disk partition to store user programs that are not currently being used. Typically, page sizes are 4 KB or 8 KB. In Linux, the page size is defined in the kernel header file include/asm-<architecture>/param.h using the variable EXEC_PAGESIZE. The process used to page a process out to disk is called *pageout*.

3.3.1 Finding memory bottlenecks

Start your analysis by listing the applications that are running on the server. Determine how much physical memory and swap each of the applications needs to run. Figure 3-1 on page 57 shows KDE System Guard monitoring memory usage.



90 Processes

Memory: 225,804 KB used, 289,340 KB free

Swap: 0 KB used, 1,048,120 KB free

Figure 3-1 KDE System Guard memory monitoring

The indicators shown in Table 3-1 can also help you define a problem with memory.

Table 3-1 Indicator for memory analysis

Memory indicator	Analysis
Memory available	This indicates how much physical memory is available for use. If, after you start your application, this value has decreased significantly, you may have a memory leak. Check the application causing it and make the necessary adjustments. Use **`free -l -t -o`** for additional information.
Page faults	There are two types of page faults: soft page faults, when the page is found in memory, and hard page faults, when the page is not found in memory and must be fetched from disk. Accessing the disk will slow your application down considerably. The **`sar -B`** command can provide useful information for analyzing page faults, specifically columns pgpgin/s and pgpgout/s.
File system cache	This is the common memory space that k the **`free -l -t -o`** command, for example.
Private memory for process	This represents the memory used by each process running on server. You can see how much memory is allocated to specific process using the **`pmap`** command.

Paging and swapping indicators

In Linux, as with all UNIX-based operating systems, there are differences between paging and swapping. Paging moves individual pages to swap space on the disk; swapping is a bigger operation which moves the entire address space of a process to swap space in one operation.

Swapping can have one of two causes:

- A process enters sleep mode. This normally happens because the process depends on interactive action, because editors, shells, and data entry applications spend most of their time waiting for user input. During this time, they are inactive.

- A process behaves poorly. Paging can be a serious performance problem when the amount of free memory pages falls below the minimum amount specified, because the paging mechanism is not able to handle the requests for physical memory pages and the swap mechanism is called to free more pages. This significantly increases I/O to disk and will quickly degrade a server's performance.

If your server is always paging to disk (a high page-out rate), consider adding more memory. However, for systems with a low page-out rate, it may not be affecting performance.

3.3.2 Performance tuning options

It you believe there is a memory bottleneck, consider performing one or more of these actions:

- Tune the swap space using bigpages, hugetlb, shared memory.
- Increase or decrease the size of pages.
- Improve the handling of active and inactive memory.
- Adjust the page-out rate.
- Limit the resources used for each user on the server.
- Stop the services that aren't needed as discussed in 1.1, "Disabling daemons" on page 2.
- Add memory.

3.4 Disk bottlenecks

The disk subsystem is often the most important aspect of server performance and is usually the most common bottleneck. However, problems can be hidden by other factors, such as lack of memory. Applications are considered to be "I/O bound" when CPU cycles are wasted simply waiting for I/O tasks to finish.

The most common disk bottleneck is having too few disks. Most disk configurations are based on capacity requirements, not performance. The least expensive solution is to purchase the smallest number of the largest-capacity disks possible. However, this places more user data on each disk, causing greater I/O rates to the physical disk and allowing disk bottlenecks to occur.

The second most common problem is having too many logical disks on the same array. This increases seek time and greatly lowers performance.

The disk subsystem is discussed in 1.9, "Tuning the file system" on page 16.

It is recommended that you apply the diskstats-2.4.patch to fix problems with disk statistics counters which can occasionally report negative values.

3.4.1 Finding disk bottlenecks

A server exhibiting the following symptoms may be suffering from a disk bottleneck (or a hidden memory problem):

- Slow disks will result in memory buffers filling with write data (or waiting for read data), which will delay all requests because free memory buffers are unavailable for write requests (or the response is waiting for read data in the disk queue), *or* insufficient memory, as in the case of not having enough memory buffers for network requests, will cause synchronous disk I/O.
- Disk and/or controller utilization will typically be very high.

- Most LAN transfers will happen only after disk I/O has completed, causing very long response times and low network utilization.
- Because disk I/O can take a relatively long time, and disk queues will become full, the CPUs will be idle or have low utilization as they wait long periods of time before processing the next request.

The disk subsystem is perhaps the most challenging subsystem to properly configure. Besides looking at raw disk interface speed and disk capacity, it is key to also understand the workload: Is disk access random or sequential? Is there large I/O or small I/O? Answering these questions will provide the necessary information to make sure the disk subsystem is adequately tuned.

Disk manufacturers tend to showcase the upper limits of their drive technology's throughput. However, taking the time to understand the throughput of your workload will help you understand what true expectations to have of your underlying disk subsystem.

Table 3-2 Exercise showing true throughput for 8K I/Os for different drive speeds

Disk speed	Latency	Seek time	Total random access time[a]	I/Os per second per disk[b]	Throughput given 8 KB I/O
15,000 RPM	2.0 ms	3.8 ms	6.8 ms	147	1.15 MBps
10,000 RPM	3.0 ms	4.9 ms	8.9 ms	112	900 KBps
7,200 RPM	4.2 ms	9 ms	13.2 ms	75	600 KBps

a. Assuming that the handling of the command + data transfer < 1 ms, total random access time = latency + seek time + 1 ms.

b. Calculated as 1/total random access time.

Random read/write workloads usually require several disks to scale. The bus bandwidths of SCSI or Fibre Channel are of lesser concern. Larger databases with random access workload will benefit from having more disks. Larger SMP servers will scale better with more disks. Given the I/O profile of 70% reads and 30% writes of the average commercial workload, a RAID-10 implementation will perform 50-60% better than a RAID-5.

Sequential workloads tend to stress the bus bandwidth of disk subsystems. Pay special attention to the number of SCSI buses and Fibre Channel controllers when maximum throughput is desired. Given the same number of drives in an array, RAID-10, RAID-0 and RAID-5 all have similar streaming read and write throughput.

There are two ways to approach disk bottleneck analysis: real-time monitoring and tracing.

- Real-time monitoring must be done while the problem is occurring. This may not be practical in cases where system workload is dynamic and the problem is not repeatable. However, if the problem is repeatable, this method is very flexible because of the ability to add objects and counters as the problem becomes well understood.
- Tracing is the collecting of performance data over time to diagnose a problem. This is a good way to perform remote performance analysis. Some of the drawbacks include the potential for having to analyze large files when performance problems are not repeatable, and the potential for not having all the key objects and/or parameters in the trace and having to wait for the next time the problem occurs for the additional data.

vmstat command

One way of tracking disk usage on a Linux system is by using the `vmstat` tool. The columns of interest in `vmstat` with respect to I/O are the *bi* and *bo* fields. These fields monitor the movement of blocks in and out of the disk subsystem. Having a baseline is key to being able to identify any changes over time.

Example 3-2 vmstat output

```
[root@x232 root]# vmstat 2
r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 2  1      0   9004  47196 1141672    0    0     0   950  149    74 87 13  0  0
 0  2      0   9672  47224 1140924    0    0    12 42392  189    65 88 10  0  1
 0  2      0   9276  47224 1141308    0    0   448     0  144    28  0  0  0 100
 0  2      0   9160  47224 1141424    0    0   448  1764  149    66  0  1  0 99
 0  2      0   9272  47224 1141280    0    0   448    60  155    46  0  1  0 99
 0  2      0   9180  47228 1141360    0    0  6208 10730  425   413  0  3  0 97
 1  0      0   9200  47228 1141340    0    0 11200     6  631   737  0  6  0 94
 1  0      0   9756  47228 1140784    0    0 12224  3632  684   763  0 11  0 89
 0  2      0   9448  47228 1141092    0    0  5824 25328  403   373  0  3  0 97
 0  2      0   9740  47228 1140832    0    0   640     0  159    31  0  0  0 100
```

iostat command

There could also be performance problems encountered when too many files are opened and being read and written to, and then closed repeatedly. This could become apparent as seek times (the time it takes to move to the exact track where the data is stored) start to increase. Using the **iostat** tool, you can monitor the I/O device loading in real time. Different options allow you to drill down even further to gather the necessary data.

Example 3-3 shows a potential I/O bottleneck on the device /dev/sdb1. This output shows average wait times (`await`) of around 2.7 seconds and service times (`svctm`) of 270 ms.

Example 3-3 Sample of an I/O bottleneck as shown with iostat 2 -x /dev/sdb1

```
[root@x232 root]# iostat 2 -x /dev/sdb1

avg-cpu:  %user   %nice    %sys   %idle
          11.50    0.00    2.00   86.50

Device:    rrqm/s wrqm/s   r/s   w/s  rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await  svctm  %util
/dev/sdb1  441.00 3030.00  7.00 30.50 3584.00 24480.00  1792.00 12240.00   748.37
101.70 2717.33 266.67 100.00

avg-cpu:  %user   %nice    %sys   %idle
          10.50    0.00    1.00   88.50

Device:    rrqm/s wrqm/s   r/s   w/s  rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await  svctm  %util
/dev/sdb1  441.00 3030.00  7.00 30.00 3584.00 24480.00  1792.00 12240.00   758.49
101.65 2739.19 270.27 100.00

avg-cpu:  %user   %nice    %sys   %idle
          10.95    0.00    1.00   88.06

Device:    rrqm/s wrqm/s   r/s   w/s  rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await  svctm  %util
/dev/sdb1  438.81 3165.67  6.97 30.35 3566.17 25576.12  1783.08 12788.06   781.01
101.69 2728.00 268.00 100.00
```

The **iostat -x** (for extended statistics) command provides low-level detail of the disk subsystem. Some things to point out:

- %util: percentage of CPU consumed by I/O requests
- svctm: average time required to complete a request, in milliseconds
- await: average amount of time an I/O waited to be served, in milliseconds
- avgqu-sz: average queue length
- avgrq-sz: average size of request
- rrqm/s: the number of read requests merged per second that were issued to the device
- wrqms: the number of write requests merged per second that were issued to the device

For a more detailed explanation of the fields, see the man page for iostat(1).

Changes made to the elevator algorithm as described in “Tune the elevator algorithm” on page 20, will be seen in the avgrq-sz (average size of request) and avgqu-sz (average queue length). As the latencies are lowered by manipulating the elevator settings, the avgrq-sz will go down. You can also monitor the rrqm/s and wrqm/s to see the effect on the number of merged reads and writes that the disk can manage.

3.4.2 Performance tuning options

After verifying that the disk subsystem is a system bottleneck, a number of solutions are possible. These solutions include the following:

- If the workload is of a sequential nature and it is stressing the controller bandwidth, the solution is to add a faster disk controller. However, if the workload is more random in nature, the bottleneck is likely to involve the disk drives, and adding more drives will improve performance.
- Add more disk drives in a RAID environment. This spreads the data across multiple physical disks and improves performance for both reads and writes. This will increase the number of I/Os per second. Also, use hardware RAID instead of the software implementation provided by Linux. If hardware RAID is being used, the RAID level is hidden from the OS.
- Offload processing to another system in the network (users, applications, or services).
- Add more RAM. Adding memory will increase system memory disk cache, which in effect improves disk response times.

3.5 Network bottlenecks

A performance problem in the network subsystem can be the cause of many problems, such as a kernel panic. To analyze these anomalies to detect network bottlenecks, each Linux distribution includes traffic analyzers.

3.5.1 Finding network bottlenecks

We recommend KDE System Guard because of its graphical interface and ease of use. The tool is also available on the distribution CDs. The tool is discussed in detail in 2.7, “KDE System Guard” on page 39. Figure 3-2 shows the tool in action.

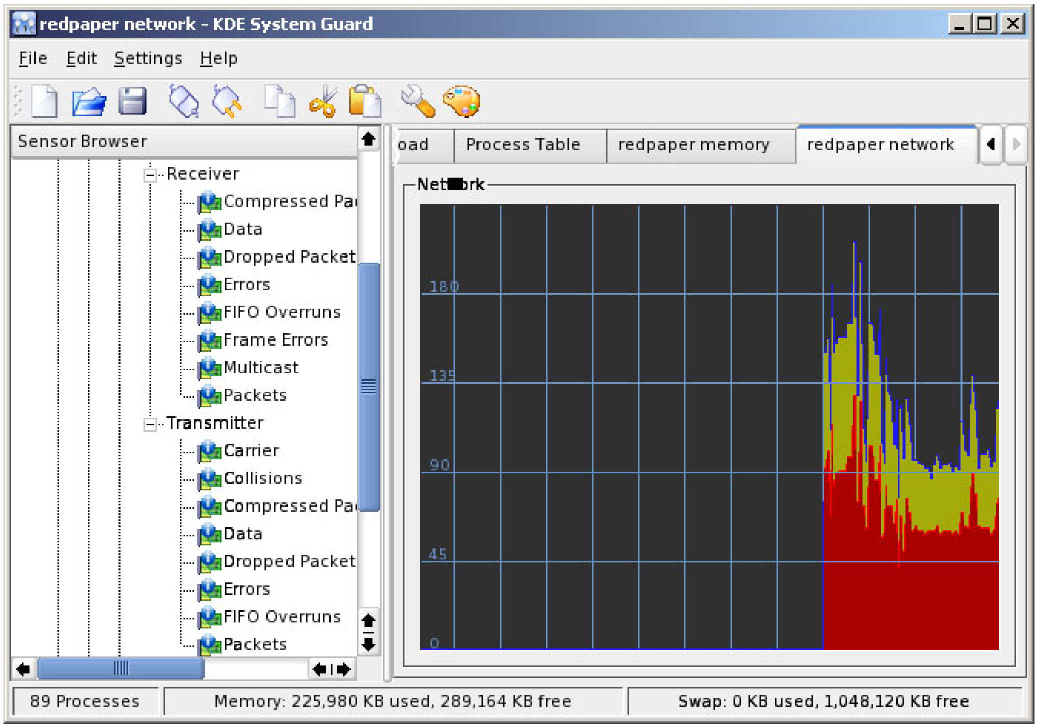


Figure 3-2 KDE System Guard network monitoring

For SUSE Linux, you can also use the **`traffic-vis`** package, an excellent network monitoring tool. You capture the network data then analyze the results using a Web browser. For details, see 2.9, “Traffic-vis” on page 44.

It is important to remember that there are many possible reasons for these performance problems and that sometimes, problems occur simultaneously, making it even more difficult to pinpoint the origin. The indicators in Table 3-3 can help you determine the problem with your network:

Table 3-3 Indicators for network analysis

Network indicator	**Analysis**
Packets received Packets sent	Shows the number of packets that are coming in and going out of the specified network interface. Check both internal and external interfaces.
Collision packets	Collisions occur when there are many systems on the same domain. The use of a hub may be the cause of many collisions.
Dropped packets	Packets may be dropped for a variety of reasons, but the result may impact performance. For example, if the server network interface is configured to run at 100 Mbps full duplex, but the network switch is configured to run at 10 Mbps, the router may have an ACL filter that drops these packets, for example: `iptables -t filter -A FORWARD -p all -i eth2 -o eth1 -s 172.18.0.0/24 -j DROP`
Errors	Errors occur if the communications lines (for instance, the phone line) are of poor quality. In these situations, corrupted packets must be resent, thereby decreasing network throughput.

Network indicator	Analysis
Faulty adapters	Network slowdowns often result from faulty network adapters. When this kind of hardware fails, it may begin to broadcast junk packets on the network.

3.5.2 Performance tuning options

The steps that follow illustrate what you should do to solve problems related to network bottlenecks:

- Ensure that the network card configuration matches router and switch configurations (for example, frame size).
- Modify how your subnets are organized.
- Use faster network cards.
- Tune the appropriate IPV4 TCP kernel parameters. See Chapter 1, “Tuning the operating system” on page 1. Some security-related parameters can also improve performance, as described in that chapter.
- If possible, change network cards and recheck performance.
- Add network cards and bind them together to form an adapter team, if possible.

4

Tuning Apache

This chapter will help you get the most of your Apache server. The Apache HTTP Server Project Web site, http://httpd.apache.org, gives access to versions of Apache running on almost all operating systems including Linux.

This chapter includes recommendations for tuning and optimizing Apache 2. 0.

Customizing the Apache Web server includes modification of the configuration file. In addition, Web administrators can go one step further and recompile the source code for their platform using the appropriate switch and modules. This topic will also be covered later in the chapter.

The goal of this chapter is to provide Web administrators with a view of the changes they can make to modify the performance of their Apache server. These changes involve three tasks:

1. Gathering a baseline for the Web server
2. Modifying a configuration parameter
3. Measuring and quantifying the performance change

Performance optimization is the result of multiple iterations of the last two steps until the server reaches a stable performance, hopefully in line with your goal. It is important to note that the best tuning strategy will never make up for a lack of hardware resources. It is possible to stretch the number of requests per second that you are getting from the Web server, but if the network bandwidth is your bottleneck, there is little that can be done from the application level.

This chapter includes the following topics:

4.1 Gathering a baseline

The best way to measure the performance impact of each setting change is to compare the new results with a baseline set of results. You should first take the time to measure and document your current server performance. This crucial step enables you to quantify your tuning steps. This can be done using one of the benchmarks for Web servers available free of charge on the market:

- WebBench: `http://etestinglabs.com/benchmarks/Webbench/Webbench.asp`
- WebStone: `http://www.mindcraft.com/Webstone/`
- Web server stress tool: `http://Web-server-tools.com/tools/WebStress/Webstress.htm`

You should look for at least two different parameters when measuring a baseline for your Web server:

- Throughput

  Throughput for Web servers can be measured using two different units. In both cases, the larger the number, the better.

  – Requests per second: this is usually the first number you should review when benchmarking a Web server.

  – Bits transmitted per second: this information is the bandwidth of your Web server, and will tell you if the Web server is saturating the wire.

- Latency

  This is the time that elapses between the client request being made and the results starting to come in from the Web server. If your network is in good condition, an increase in the latency will tell you that your Web server is overloaded and is having problems keeping up. A smaller value would be better.

Each of these benchmarks gives you the opportunity to test different types of requests (static, dynamic, and secure) and compute numbers to be able to define the current limits of your system (except for WebStone, which does not support SSL requests).

Having gathered a baseline, you should set a goal for your system. Why do so?

Tuning a system can be a time-consuming task. However, most of the benefits are usually gained by following general tuning steps and it is easy to measure a performance improvement. The next step is to perform tuning specific to your server's load characteristics, adjusting parameters to find the best setup for a specific server in a specific environment. This step often takes the longest time to complete.

4.2 Web server subsystems

As with other applications, running your Apache Web server on dedicated hardware is the first step, both for security and performance purposes. Apache is a lightweight server and its processes, or threads, do not consume excessive memory. The way the Web server is affected by each subsystem depends on the type of content it serves:

- Mainly static pages:

  a. Network
  b. Memory
  c. CPU

- Dynamic content:

  a. Memory
  b. CPU
  c. Disk
  d. Network

- Secure content:

  a. CPU
  b. Memory
  c. Disk
  d. Network

Usually, Apache will run out of memory before anything else when servicing mixed content. The amount of memory significantly influences the performance of your Web server. The more memory you have, the more the server can cache the data requested and serve it to users faster than if the data had been on disk.

So how do you know whether you are using enough memory in your server? There is no straight answer to this question. You need enough memory to run all the different applications, and enough memory to cache and process the most requested files. You will be able to gauge the amount of memory for this task by observing and analyzing the server in its real environment with the different tools described in Chapter 2, “Tuning tools” on page 31.

Apache works best when it does not need to use the swap partition.

CPU capacity depends greatly on the content being served, as described above. Figure 4-1 shows the CPU load serving static pages (plain HTML) versus the CPU load serving dynamic pages (a CGI program written in C, in our example).

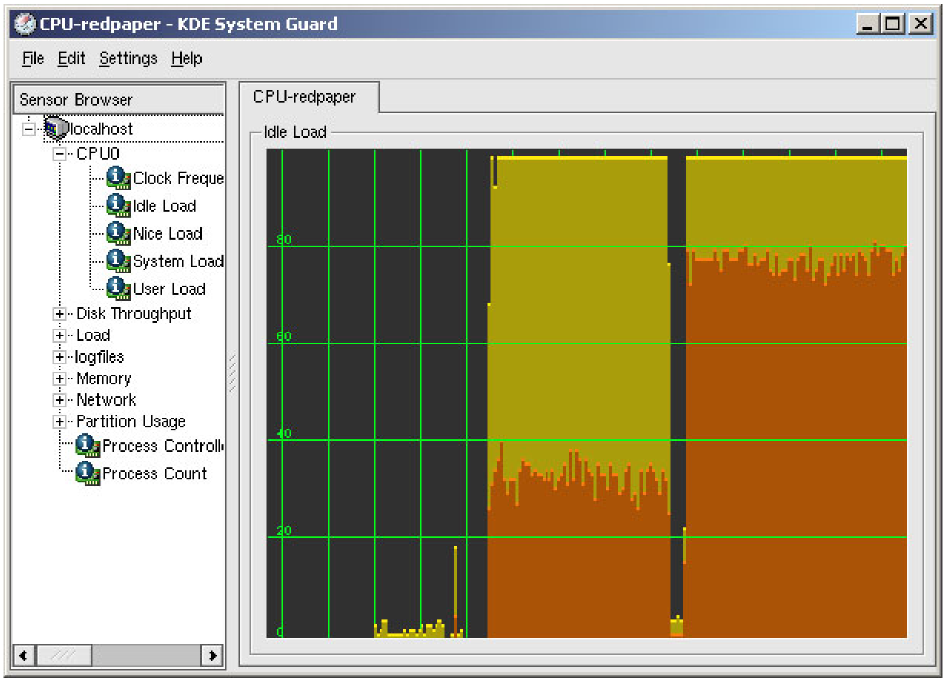


Figure 4-1 Static pages (left) and dynamic pages (right) in CPU utilization

4.3 Apache architecture models

Two architecture models are supported by Apache 2.0:

- Process-driven architecture model

  The process-driven (or *fork*) architecture creates a separate process to handle each connection. Each new process is a copy of the original process. When started, Apache creates several new child processes to handle Web server requests in addition to always keeping a set number of processes idle to handle peak demand (Figure 4-2).

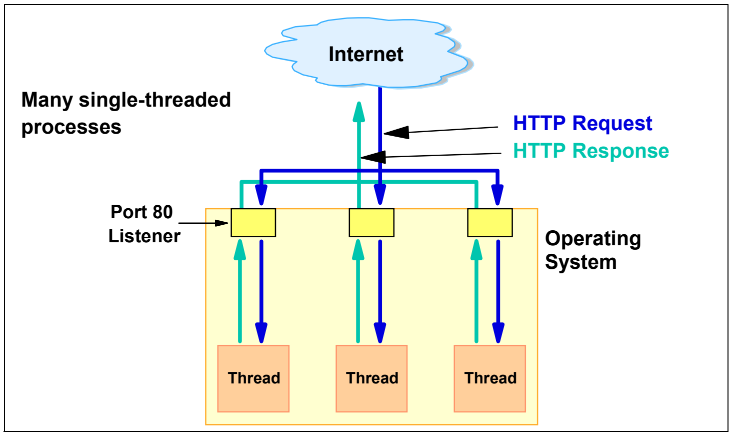


Figure 4-2 Web server based on a process-driven model

- Multi-threaded architecture model

  Apache V2.0 offers the option of using a second model, the multi-threaded architecture. According to the Apache Foundation, it should improve scalability for many configurations.

  With this model, only two processes are created to handle all requests. Within one of the processes, threads are created to handle server requests.

  A thread (also called a lightweight process) is a stream of control that can execute its instructions independently. More simply put, a thread is a unit of execution sharing resources with other threads within a process (Figure 4-3).

  The multi-threaded model is theoretically a much more efficient architecture implementation.

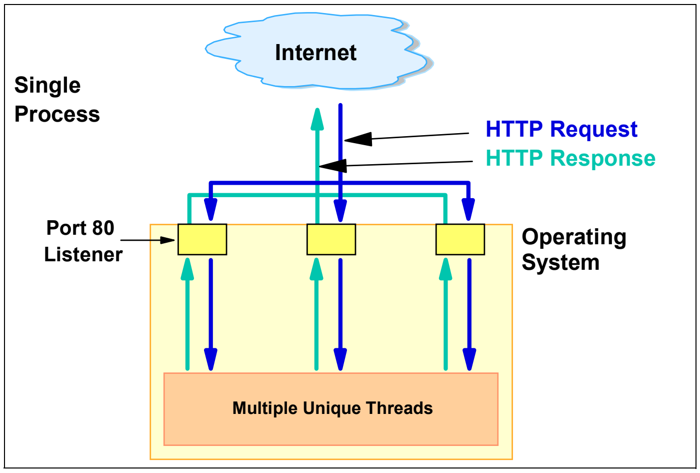


Figure 4-3 Web server based on a multi-threaded model

4.4 Compiling the Apache source code

Compiling the application offers performance advantages. Compiling Apache with only the needed modules is a crucial step in making sure that Apache is not spending precious CPU cycles and memory bytes on unused modules. If Apache was installed using the normal installation process, it is running from the default binaries. Compiling the code enables you to build and run a suitable version for your system.

See the following Web site for information about how to compile the Apache source code:

http://httpd.apache.org/docs-2.0/install.html

> **Restriction:** Recompiling Apache or any other package (including Apache modules like php or Perl) voids standard SUSE service and maintenance liabilities for these packages and anything using them. You should contact SUSE LINUX to request an individual service agreement on such a non-standard setup.

4.5 Operating system optimizations

A variety of operating system tuning parameters are relevant to Apache.

The maximum file handles that Linux supports will impact how many pages Apache can serve simultaneously. The following command displays the current maximum:

```
linux:~ # sysctl fs.file-max
fs.file-max = 52427
```

An acceptable value may be closer to 256 KB. To set this value, use the following command:

```
sysctl -w fs.file-max=262144
```

The file /etc/security/limits.conf lets you specify a variety of limits:

- How many processes and child processes a user can open
- How much memory a user can consume using soft and hard limits
- Maximum CPU time
- Maximum size locked in memory address space
- Maximum stack size

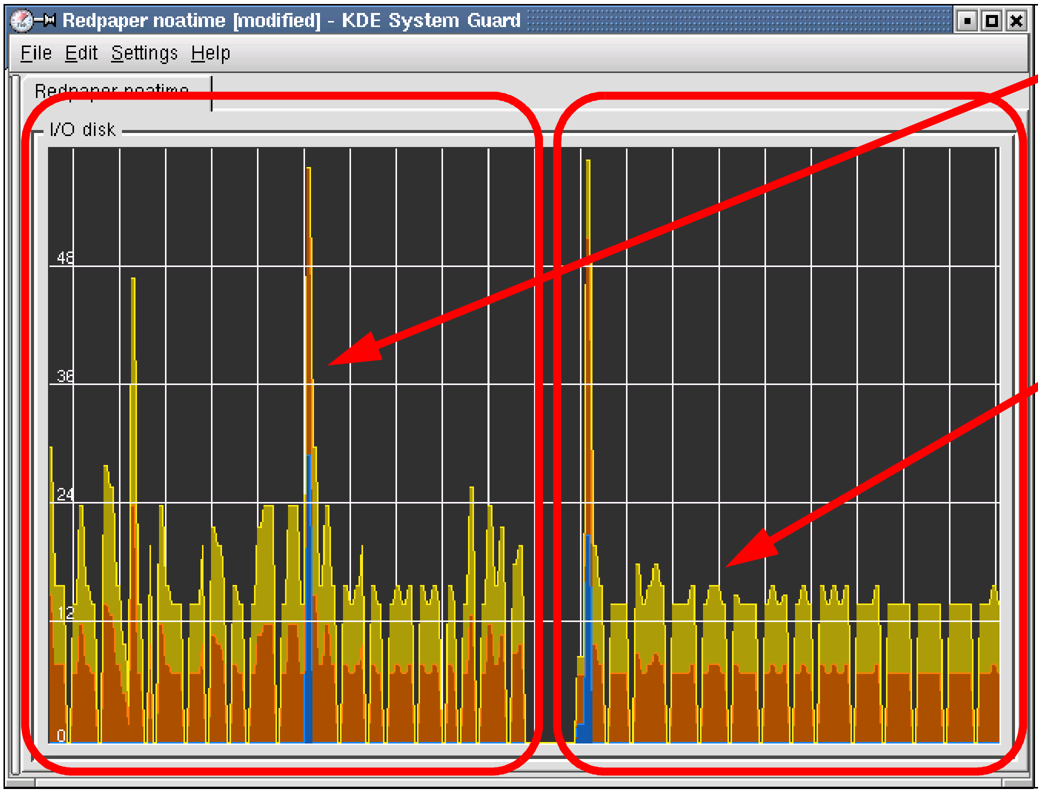
For example, to set the maximum number of processes a user can open, add these lines to the file /etc/security/limits.conf:

```
soft    nproc 4096
hard    nproc 8192
```

Assuming you are using the bash shell, to see what kind of system resource a user can consume, use the **ulimit** command as described in 2.12, “ulimit” on page 48:

```
ulimit -a
```

Linux records when a file was last modified or accessed; there is a cost associated with this. In the ReiserFS file system, disabling this feature may lead to significant performance improvements. See Figure 4-4 on page 70 for an example of I/O throughput when the parameter is enabled (left) and disabled (right).



By default, `atime` (access time) is enabled, resulting in performance spikes.

With `noatime` specified in /etc/fstab, no file accesses are logged so disk performance is smooth.

Figure 4-4 atime enabled (left) and atime disabled (right)

To disable this feature, put the option `noatime` into the /etc/fstab on the line related to the file system. This is the entry before the change:

```
/dev/sda2            /                    reiserfs   defaults             1 1
```

This is the entry after the change. Put the `noatime` parameter after the defaults parameter, separated by a comma:

```
/dev/sda2            /                    reiserfs   defaults,noatime 1 1
```

Important: The atime parameter is used by back-up software to determine what has changed in order to perform an incremental back-up. If you disable atime, incremental back-ups will in effect perform a full back-up each time.

4.6 Apache 2 optimizations

If you make changes to Apache directives, normally you should restart Apache before they take effect. Apache 2.0 has a “hot restart” feature that enables you to apply configuration changes to a server while it is still servicing requests. None of the connections is broken, and clients should not notice any interruption of service.

See `http://httpd.apache.org/docs-2.0/stopping.html` for details.

Apache uses the file httpd.conf to store configuration parameters known as directives. More information about these directives can be found at:

```
http://httpd.apache.org/docs/mod/core.html
http://httpd.apache.org/docs-2.0/mod/core.html
```

Directives that are relevant to performance tuning include the following:

- Timeout

  This directive is set, by default, to 300 seconds. This is the maximum delay Apache allows an HTTP connection to remain open after the last interaction. This value is excessively high because no user will ever wait five minutes to complete a request. Our recommendation is to reduce it until valid connections do not time out:

  ```
  Timeout 60
  ```

- KeepAlive

  The default value is *on* and we recommend that you keep it this way.

  ```
  KeepAlive on
  ```

  This parameter allows for persistent connections, multiple sequential requests from the client inside the same TCP connection, allowing a much faster dialog between the client and the server. In addition to being faster, KeepAlive reduces traffic on the link by removing the connection negotiation for each request:

  KeepAlive will provide your users with a huge performance improvement by decreasing the latency between requests. The latency could be cut by two thirds if your server is not overloaded. With a server using most of its CPU capacity, your gain will be twofold:

  – Serving more clients in the same time interval because more CPU cycles are available and network bandwidth is less utilized

  – Faster response to your users

  For a loaded server, you should easily see a gain of around 50% in the number of requests per second.

- MaxKeepAliveRequests

  The default value is usually equal to 100. This value limits the number of HTTP requests for which a single TCP connection will stay alive. Persistent connections will automatically be ended after that value is reached and connection negotiation will need to be restarted after this point.

  A high value is preferable but needs to be set in conjunction with the KeepAliveTimeOut parameter to be able to clean up the dead connections at regular intervals. This value could also be set to 0, allowing an unlimited number of requests within a single connection.

  We recommend setting this value to as high a number as possible, especially if your users are habitually requesting many files in the same session.

  ```
  MaxKeepAliveRequests 400
  ```

  When this parameter reaches its limit, the TCP connection will terminate and will need to be reinitiated from the client browser.

- KeepAliveTimeOut

  This parameter sets the maximum time Apache will wait between two requests before ending the connection. The default value is 15 seconds. Whether or not to change this value depends on your network speed and traffic. If multiple browsers are not properly closing the KeepAlive connections with the server, these connections will stay unavailable for other clients during this period. On the other hand, for a slow connection (modem for example), the time-out value may need to be increased or each request will have to go through the connection process again and again during a single session.

  ```
  MaxAliveTimeOut 15
  ```

When this parameter reaches its limit, the TCP connection will terminate and will need to be reinitiated from the client browser.

- DNS resolution (HostnameLookups)

  This directive is set to *off* by default in V2.0 and V1.3, but was on by default for earlier versions. Setting this directive to `on` will give you the DNS name of the browser performing a request to your server instead of using only the IP address. In addition to generating network traffic, setting this directive to `on` will add latency.

  ```
  HostnameLookups off
  ```

  If you set this directive to `off`, a log entry will look like this:

  ```
  137.65.67.59 - - [24/May/2002:09:38:16 -0600 "GET /apache_pb.gif HTTP/1.1"
  200 2326
  ```

  If you set this directive to `on`, the log entry will look like this:

  ```
  furby.provo.novell.com - - [24/May/2002:09:37:54 -0600 "GET /apache_pb.gif
  HTTP/1.1" 200 2326
  ```

  If your log analysis requires the resolution of IP addresses, consider using a tool such as **`logresolve`** to perform this translation. For more information, visit:

  http://httpd.apache.org/docs-2.0/programs/logresolve.html

- AllowOverride

  This directive specifies whether the .htaccess file is to be read to determine access authorities. To prevent this file from being read, thereby improving performance, use this directive:

  ```
  AllowOverride None
  ```

- Use of sendfile

  Apache normally uses sendfile to supply static files to the Web client. However, if the files are stored on an NFS file system, the files are not cached by Apache so performance may suffer. If using NFS, consider disabling the directive:

  ```
  EnableSendfile Off
  ```

- Extended Status

  Apache uses the module mod_status to provide a Web page to administrators that shows the server's current status. This includes the number of requests being processed currently and the number of idle child processes. The ExtendedStatus directive can provide additional information (such as the total number of accesses and the total bytes served).

  If your Apache installation is not using mod_status to monitor child processes and the status between them, ensure ExtendedStatus is also disabled (the default is off) because every request Apache will need to call gettimeofday for timing information:

  ```
  ExtendedStatus Off
  ```

4.6.1 Multi-processing module directives

The following directives relate to the mpm module:

- StartServer

  This is the number of child processes Apache will create on startup. This setting only applies at startup. After startup, the number of child processes is dynamically controlled and is dependent on other settings. If you have to restart your server frequently, or while servicing requests, it is a good idea to increase this number so the server will get back up to speed quickly.

  The default value is 5, but should be set close to the average number of child processes while under normal load to minimize startup delay times. The algorithm to create new processes in Apache is to use a minimum delay of one second before creating a new process, and then the number of processes created is doubled every second until it reaches 32 processes per second or until the load can be handled without having to create new processes.

  ```
  StartServer 5
  ```

- MinSpareServers

  This setting specifies the minimum number of idle child processes that must be available to service new connections at any given time. The pool of available processes is updated by Apache to remain at least equal to that limit when the connection number increases. These processes are useful when you are experiencing spikes.

  The default value for this parameter is 5. For heavily used sites experiencing a lot of spikes, it should be increased to 25. This will reduce the latency time users are experiencing when connecting during a climbing load period.

  ```
  MinSpareServers 10
  ```

- MaxSpareServers

  This setting defines the maximum number of idle child processes that can be made available to service new connections at any given time. This pool of processes is updated by Apache to remain between the minimum and the maximum values when the connection number increases or decreases. Having too many idle processes, however, is a waste of resources. If the number of processes that are available in the idle pool exceeds this value, Apache will terminate the excess processes.

  The default value for this parameter is 20, but for a busy server with enough memory available, it could be set to 100-125.

  ```
  MaxSpareServers 50
  ```

- MaxClients

  This parameters defines the maximum number of child processes available simultaneously. That is, it is the maximum number of requests that will be served at any one time.

  The default value is 150 and the maximum value that can be used without recompiling the server binaries is 256. If you use a value higher than 256, the server will load but warn you and set the maximum number of clients to 256.

If your site includes many dynamic pages and you do increase this value, you may start to experience memory problems and access to swap. In these circumstances, you should consider reducing the value. The incoming requests, instead of being processed immediately, will be put on a queue until the process becomes available. This will work faster than having to swap memory to disk. However, a better solution would be to increase the amount of RAM installed.

```
MaxClients 100
```

- MaxRequestsPerChild

  This directive controls how many requests a process will serve before exiting. The default, 0, causes the process never to exit. However, accepting this value of 0 could lead the system to memory leaks if you use poorly written modules. On the other hand, ending and restarting processes is an expensive set of operations. If you accept this value of 0, keep monitoring the memory utilization to determine if memory leaks are occurring.

  ```
  MaxRequestsPerChild 0
  ```

Figure 4-5 illustrates the different parameters for setting the number of active processes.

Note: The relationship between these parameters is as follows:

```
MinSpareServers < number of spare processes (Sp) < MinSpareServers

Total number of process (Tp) = number of connections + Sp

Tp <= MaxClients
```

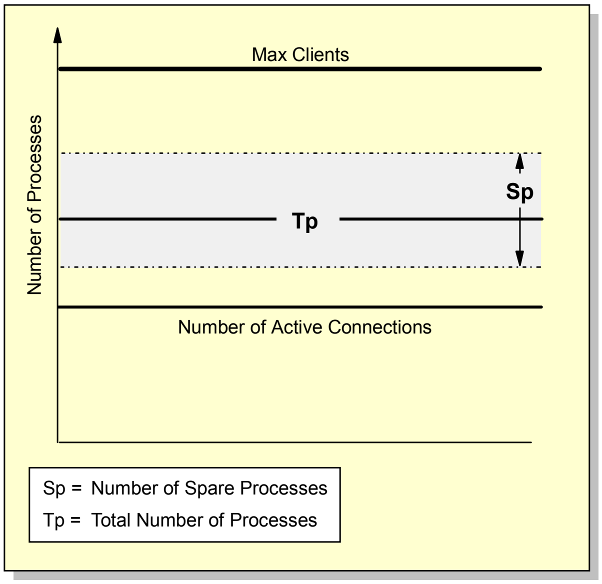

Max Clients
Number of Processes
Sp
Tp
Number of Active Connections
Sp = Number of Spare Processes
Tp = Total Number of Processes

Figure 4-5 Process limits in a process-driven architecture model

4.6.2 Compression of data

Compression of text files can reduce their sizes significantly, thereby reducing networking costs and improving performance. For example, reductions of 72% are possible using compression level 6 (medium). However, compression requires CPU capacity; the higher the level of compression (values 6-9 are valid), the greater the CPU requirements, so there is a trade-off.

Apache can compress the following files using GZIP-encoding:

- HTML files or files in plain/text
- Postscript files

Some files times are not (or should not be) compressed. These include:

- PDFs, which are already compressed
- JPG and GIF images, which are already compressed
- JavaScript, mainly because of bugs in browser software

Apache uses module mod_deflate to perform the compression. See Table 4-1 and Figure 4-6 on page 76 for an example of an HTML file with and without compression enabled.

- Web server: Apache/2.0.46
- Document Path: /compress_redpaper.html
- Concurrency Level: 150
- Complete requests: 50000
- Failed requests: 0
- Broken pipe errors: 0

Table 4-1 Information about Apache 2 with mod_deflate

Category	No compression	Compression with mod_deflate
Document size sent	29,139 bytes	8,067 bytes
Time taken for tests	134 seconds	485 seconds
Total transferred	1,403.44 MB	399.63 MB
HTML transferred	1,390.64 MB	384.66 MB
Requests per second (mean)	372	103
Time per request (mean)	2.69 ms	9.70 ms
Transfer rate	10,951 KBps	863 KBps

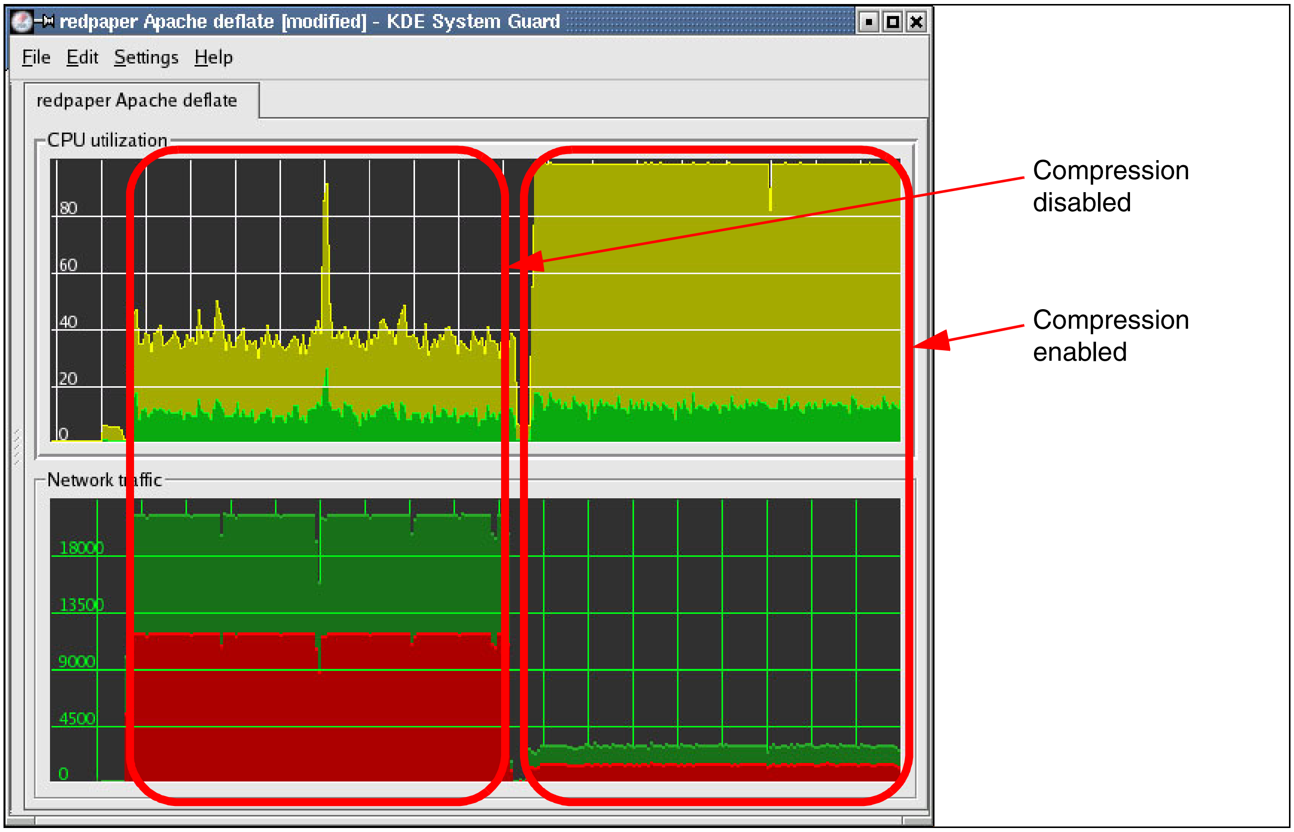


Figure 4-6 Apache 2 without data compression (left) and using data compression (right)

Analyzing the data and graphs, the following can be seen:

- Compression reduced network bandwidth by 70%.
- Compression increased CPU utilization by 87% or more. The CPU was saturated in our tests.
- With compression enabled, only a third of the number of client requests could be serviced.

You must determine the best option for your configuration based on client needs and associated server hardware requirements.

What should be compressed?

You can specify which file types are to be compressed by a directive in the httpd.conf file:

```
#Compression only HTML files
AddOutputFilterByType DEFLATE text/html text/plain text/xml
```

For all types of files (except images file), the compression will be done where we put the filter DEFLATE, such as:

```
<Location />
SetOutputFilter DEFLATE
```

Some browsers and versions of browsers have problems with the compression of specific types of files. To filter what each one should receive, use the directive **BrowserMatch**:

```
#Netscape 4.x
BrowserMatch ^Mozilla/4 gzip-only-text/html
#Netscape 4.06-4.08
```

```
BrowserMatch ^Mozilla/4\.0[678] no-gzip
#MSIE working like as Netscape
BrowserMatch \bMSIE !no-gzip !gzip-only-text/html
```

For images that do not need to be compressed, such as JPG, GIF, and PNG, compression can be disabled with these directives:

```
#Exception for images already compressed
SetEnvIfNoCase Request_URI \
\.(?:gif|jpe?g|png)$ no-gzip dont-vary
```

The **vary** directive is used to advise the proxies to send only compressed contents to clients that understand this:

```
#Make sure proxies don't deliver the wrong content
Header append Vary User-Agent env=!dont-vary
</Location>
```

Important: Apache 2 only compresses data prior to sending it to the client if the HTTP header of the client's request includes either of these:

```
Accept-encoding: gzip
Accept-encoding: gzip, deflate
```

Compression directives

These directives determine the characteristics of the mod_deflate module and its impact on server performance.

For details of all directives in mod_deflate, see:

http://httpd.apache.org/docs-2.0/mod/mod_deflate.html

- DeflateBufferSize

  Specifies the blocks of memory that should be compressed at one time. The default is 8192.

  ```
  DeflateBufferSize 16384
  ```

- DeflateCompressionLevel

  Sets the level of compression Apache should use. The default is 6. Level 9 specifies maximum compression but at great CPU cost.

  ```
  DeflateCompressionLevel 9
  ```

- DeflateMemLevel

  Defines how much memory Apache should used to compress. The default is 9.

  ```
  DeflateMemLevel 9
  ```

- DeflateWindowSize

  Specifies the zlib compression window size. The default is 15.

  ```
  DeflateWindowSize 15
  ```

4.6.3 Logging

Logging is an expensive but sometime necessary operation. Each entry has to be written to the log files, and on a busy site, this could mean thousands of entries per minute, a lot of disk space, and a number of significant CPU cycles. There are, with the default configuration, two log files available with Apache:

- Access log
- Error log

Access log

This log gets a new entry each time a request is received. This is equal to about 1 MB for every 10,000 requests. The access log is a good tool if you wish to track who is using your site and which files are mostly requested. An example of a log entry is:

```
127.0.0.1 - - [24/May/2002:12:15:11 -0700] "GET /apache_pb.gif HTTP/1.1" 304 0
```

If you want to maximize the performance of your server, you should turn access logging off by commenting out (put a `#` in front of it) the `CustomLog` entry from your configuration file. See Example 4-1 for details.

Example 4-1 Access log location in httpd.conf

```
# The location and format of the access logfile (Common Logfile Format).
# If you do not define any access logfiles within a <VirtualHost>
# container, they will be logged here. Contrariwise, if you *do*
# define per-<VirtualHost> access logfiles, transactions will be
# logged therein and *not* in this file.
#
CustomLog logs/access.log common
```

Error log

There are eight levels of logging available for the error log, as shown in Figure 4-7 on page 79. The default setting is *info*, which should give you all the information you need about your server in normal operating mode.

The error log is a helpful tool when you are experiencing problems with your Web server, but it can be costly under a high load. To minimize the logging, set the level to `error` or lower the level. If you set the level to `error`, you should see logged entries for missing files (Example 4-2). There are probably broken links in your Web site that need to be updated.

Example 4-2 Example of a missing file log entry

```
[Thu May 23 17:20:25 2002] [error] [client 10.0.10.101] File does not exist:
/usr/local/apache2/htdocs/tech/lab/test/tractor.gif
```

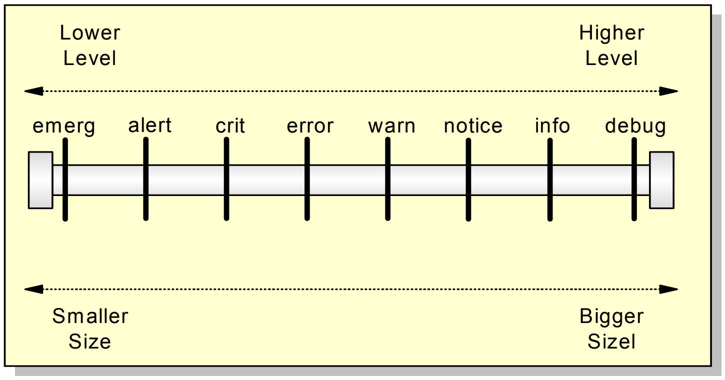


Figure 4-7 Levels of logging for the error log file

> **Tip:** If you are using WebBench as your baseline tool, you need to know that the default workload files are requesting 2% of files that are missing (Error 404). This is a small percentage, but it could increase the size of your error_log file to several MB during a single full run. Just set your logging level to a lower level than error: crit, alert or emerg.
>
> Do not forget to set it back to a lower level after you are done with WebBench.

4.6.4 Apache caching modules

Apache 2.0 includes a series of modules that bring caching capabilities to the Web server. Using caching modules could give you up to a 100% boost in the number of static files served within the same time interval.

The online documentation is available from:

http://httpd.apache.org/docs-2.0/mod/mod_cache.html

Memory caching for Apache 2.0 requires two modules, one main module, mod_cache, and one responsible for in-memory caching, mod_mem_cache.

General caching directives

The following directives apply to the main module (mod_cache):

- CacheOn

  This directive is set by default to *off*. Currently, loading the module is enough to start caching, and this directive does not need to be set.

  ```
  CacheOn On
  ```

- CacheDefaultExpire

  This is the default time in seconds that an entry will stay in cache without expiring. This default value will be used if the Expires field in the file header is 0 or invalid and the Last Modified field is invalid. The default value is 3600 seconds, which represents one hour.

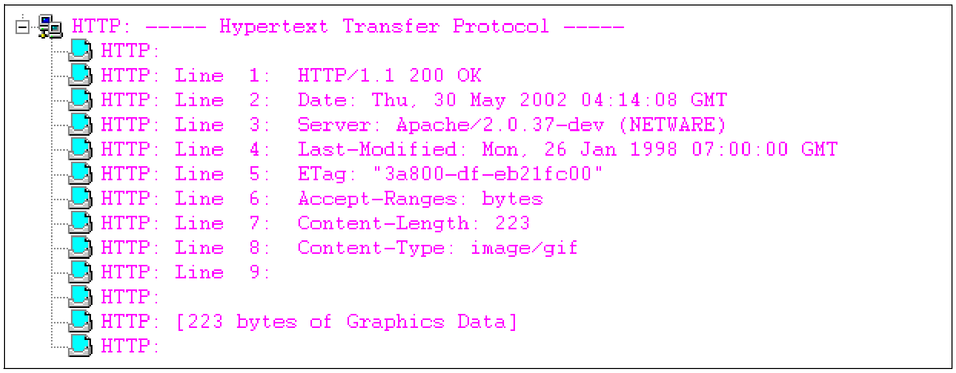
  ```
  CacheDefaultExpire 43200
  ```

- CacheMaxExpire

  This is the maximum time, in seconds, that an entry will stay in cache without expiring. This default value will be used if the Expires field in the file header is 0 or invalid and a valid Last Modified is present. The default value is 86 400 seconds, which represents 24 hours. This directive has precedence over the previous one based on the Last Modified field of the header.

  ```
  CacheDefaultExpire 252000
  ```

Figure 4-8 displays a partial header showing the Last Modified field used for the two previous cache directives. We encourage you to use the default value for CacheMaxExpire or to set it to a large interval of time, especially if your Web site is stable and does not carry file changes often. Setting that value to a small interval, such as 25 seconds, could drop the performance of your site by as much as 50%.



```
HTTP: ----- Hypertext Transfer Protocol -----
  HTTP:
  HTTP: Line  1:  HTTP/1.1 200 OK
  HTTP: Line  2:  Date: Thu, 30 May 2002 04:14:08 GMT
  HTTP: Line  3:  Server: Apache/2.0.37-dev (NETWARE)
  HTTP: Line  4:  Last-Modified: Mon, 26 Jan 1998 07:00:00 GMT
  HTTP: Line  5:  ETag: "3a800-df-eb21fc00"
  HTTP: Line  6:  Accept-Ranges: bytes
  HTTP: Line  7:  Content-Length: 223
  HTTP: Line  8:  Content-Type: image/gif
  HTTP: Line  9:
  HTTP:
  HTTP: [223 bytes of Graphics Data]
  HTTP:
```

Figure 4-8 Partial header of an Apache retransmission

- CacheEnable and CacheDisable

  These directives instruct the caching modules to allow or deny caching of URLs above the URL string pass in a parameter. In both cases, the type of caching needs to be set in the argument line (`mem` for memory in our case).

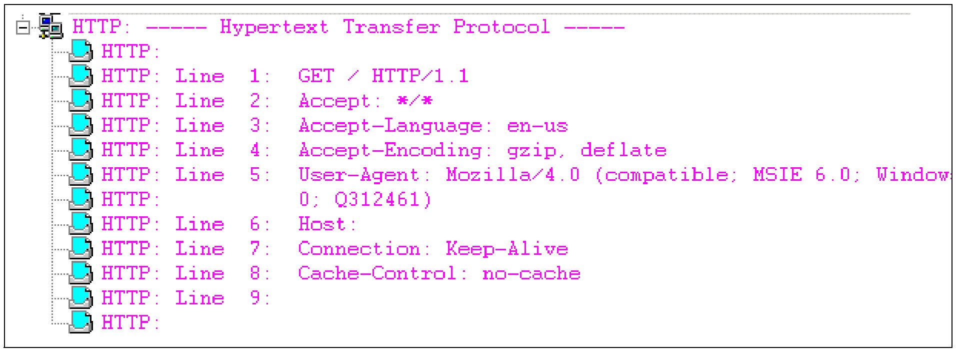
  ```
  CacheEnable mem /Webtree/tractors
  CacheDisable mem /Webtree/wheel_loaders
  ```

- CacheIgnoreCacheControl

  This directive will let you cache and serve from the cache files that the client is trying to always get fresh from the disk by using the no-cache or no-store parameter in the request. By default, this value is Off but should be changed to enforce a greater number of requests served from the cache.

  ```
  CacheIgnoreCacheControl On
  ```

  Figure 4-9 on page 81 shows an example of a GET request instructing the Web server not to cache the requested file (`Cache Control: no cache`).



```
HTTP: ----- Hypertext Transfer Protocol -----
    HTTP:
    HTTP: Line  1:  GET / HTTP/1.1
    HTTP: Line  2:  Accept: */*
    HTTP: Line  3:  Accept-Language: en-us
    HTTP: Line  4:  Accept-Encoding: gzip, deflate
    HTTP: Line  5:  User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Window
    HTTP:           0; Q312461)
    HTTP: Line  6:  Host:
    HTTP: Line  7:  Connection: Keep-Alive
    HTTP: Line  8:  Cache-Control: no-cache
    HTTP: Line  9:
    HTTP:
```

Figure 4-9 Get request instruction with no cache parameter

In-memory caching directives

The following directives apply to the memory caching module and will configure the limits of the cache.

- MCacheSize

  This is the maximum amount of memory used by the cache, in KB. The default value is 100 KB. You should first determine the size of the Web sites you are hosting, and then the amount of memory you have on your server, and finally set this directive in accordance with these values.

  ```
  MCacheSize 60000
  ```

- MCacheMaxObjectCount

  This is the maximum number of objects to be placed in the cache. If you want to cache all the static files within your Web tree, just set that to a higher value, and then set the number of files your server is servicing. The default value is 1000 objects.

  ```
  MCacheMaxObjectCount 6500
  ```

- MCacheMinObjectSize

  This is the minimum size, in bytes, that an object has to be in order to be eligible for caching. This is used if you do not want small objects to be cached and reach the maximum number of objects without filling the memory space. Having many objects in the cache could also increase the search time for each URL in the hash array. The default value is 0 bytes.

  ```
  MCacheMinObjectSize 0
  ```

- MCacheMaxObjectSize

  This is the maximum size in bytes that an object has to be to be eligible for caching. This is used if you do not want large files to be cached, in the event of a small memory situation. The default value is 10 000 bytes but should be set much higher if you do not have memory limitations.

  ```
  MCacheMaxObjectSize 580000
  ```

Example 4-3 on page 82 includes a set of directives to enable and maximize cache utilization.

Example 4-3 Example of a set of configuration for caching modules

```
LoadModule cache_module modules/mod_cache.so
<IfModule mod_cache.c>
 CacheOn On
 CacheMaxExpire 172800
 CacheIgnoreCacheControl On
 LoadModule mem_cache_module modules/mod_mem_cache.so
 <IfModule mod_mem_cache.c>
  MCacheEnable mem /
  MCacheSize 65000
  MCacheMaxObjectCount 6500
  MCacheMinObjectSize 0
  MCacheMaxObjectSize 580000
 </IfModule>
</IfModule>
```

Making sure a file is cached

How can you tell if your cache is working correctly? First, you should be able to measure a performance improvement. If you need to see details, you can change the logging level to `debug` and make at least two requests for the same file to your Apache Web server.

Example 4-4 displays the error log trace of two requests performed on the default Apache Web page (index.html which includes apache_pg.gif). The first request services the two files from the disk and caches only the GIF file because the no-cache parameter was in the request and the CacheIgnoreCacheControl was not set to `On`. As you can see, the second request will handle the GIF file from the cache.

You should also monitor your memory utilization; with caching on, it should increase.

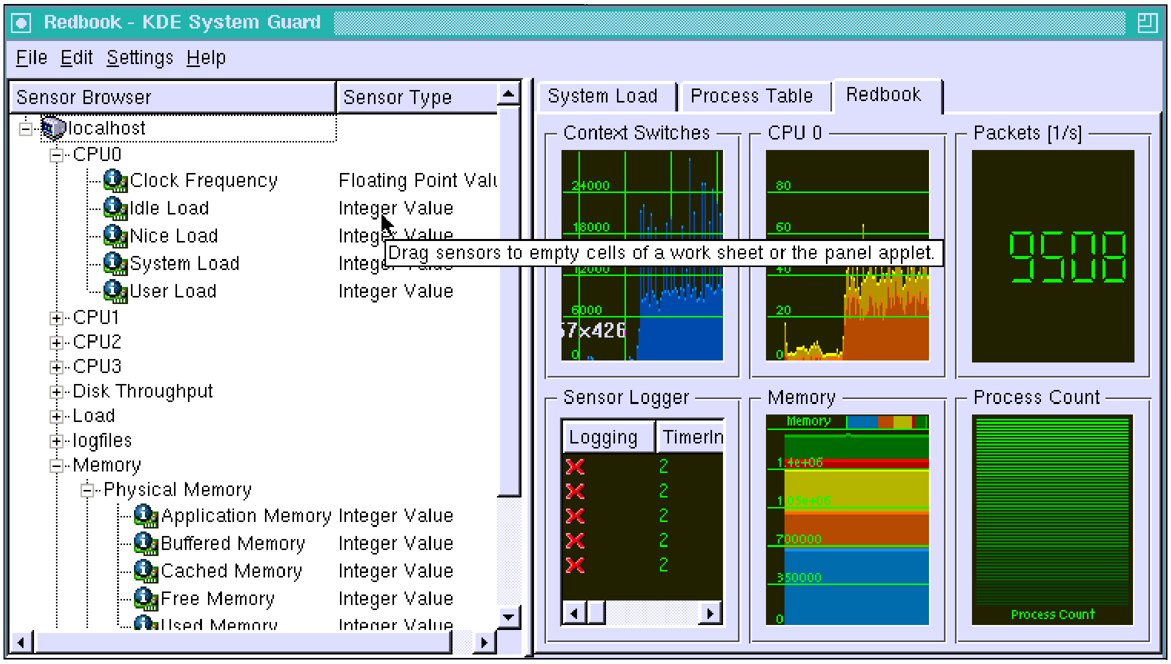
Example 4-4 Caching modules logging

```
[debug] mod_cache.c(109): cache: URL / is being handled by mem
[debug] mod_cache.c(109): cache: URL /index.html is being handled by mem
[debug] mod_cache.c(109): cache: URL /index.html.var is being handled by mem
[debug] mod_cache.c(109): cache: URL /apache_pb.gif is being handled by mem
[debug] mod_cache.c(194): cache: no cache - add cache_in filter and DECLINE
[debug] mod_cache.c(419): cache: running CACHE_IN filter
[debug] mod_cache.c(650): cache: Caching url: /apache_pb.gif
[debug] mod_cache.c(681): cache: Added date header
[debug] mod_cache.c(109): cache: URL / is being handled by mem
[debug] mod_cache.c(109): cache: URL /index.html is being handled by mem
[debug] mod_cache.c(109): cache: URL /index.html.var is being handled by mem
[debug] mod_cache.c(109): cache: URL /apache_pb.gif is being handled by mem
[debug] mod_cache.c(211): cache: fresh cache - add cache_out filter and handle request
[debug] mod_cache.c(339): cache: running CACHE_OUT filter
[debug] mod_cache.c(351): cache: serving cached version of /apache_pb.gif
```

4.7 Monitoring Apache

Apache 2.0 does not include graphical monitoring tools, but you can use the tools described that are in Chapter 2, “Tuning tools” on page 31, to monitor your Web server activities.

KDE System Guard, for example, gives you access to data on four subsystems: CPU, memory, network, and disk. You select the specific sensors you wish to monitor in the workspace area of KDE System Guard. Figure 4-10 displays part of the sensors that are available to monitor your Apache server on Linux.

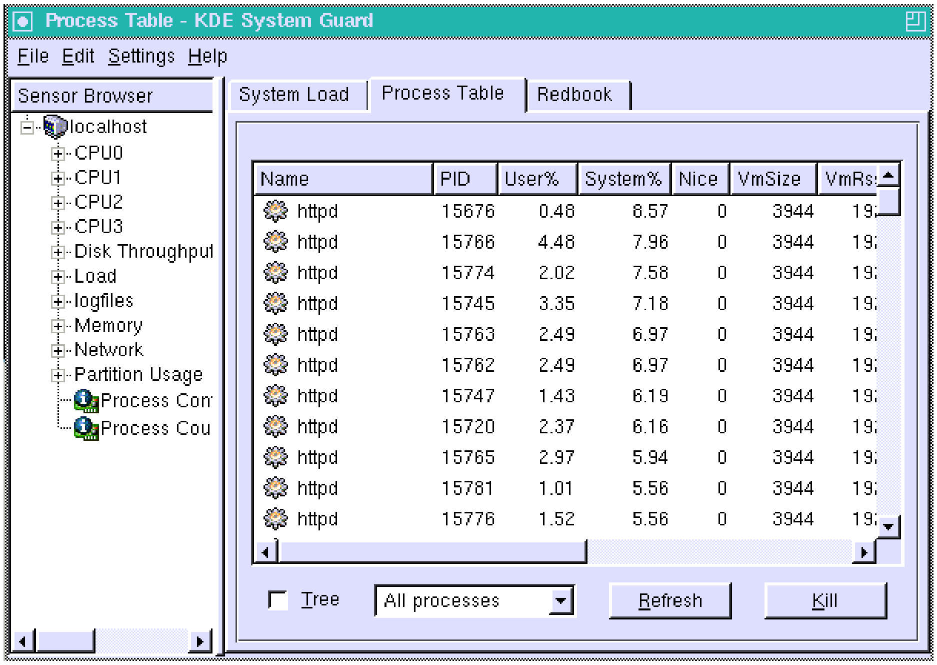


Sensor Browser	Sensor Type
localhost	
CPU0	
Clock Frequency	Floating Point Valu
Idle Load	Integer Value
Nice Load	Integer Value
System Load	Integer Value
User Load	Integer Value
CPU1	
CPU2	
CPU3	
Disk Throughput	
Load	
logfiles	
Memory	
Physical Memory	
Application Memory	Integer Value
Buffered Memory	Integer Value
Cached Memory	Integer Value
Free Memory	Integer Value
Used Memory	Integer Value

Drag sensors to empty cells of a work sheet or the panel applet.

Figure 4-10 KDE System Guard Sensor Browser for the local host

Figure 4-11 on page 84 displays information about all processes that are running on the Linux server. As you can see, the top processes when ranking the table by System% are all Apache ones. This server was under heavy load, servicing more than 3,000 requests per second when the screen capture was taken; this is why most of the CPU cycles were taken by Apache processes. Using this tool, it is also possible to kill processes.



Name	PID	User%	System%	Nice	VmSize	VmRs
httpd	15676	0.48	8.57	0	3944	19
httpd	15766	4.48	7.96	0	3944	19
httpd	15774	2.02	7.58	0	3944	19
httpd	15745	3.35	7.18	0	3944	19
httpd	15763	2.49	6.97	0	3944	19
httpd	15762	2.49	6.97	0	3944	19
httpd	15747	1.43	6.19	0	3944	19
httpd	15720	2.37	6.16	0	3944	19
httpd	15765	2.97	5.94	0	3944	19
httpd	15781	1.01	5.56	0	3944	19
httpd	15776	1.52	5.56	0	3944	19

□ Tree   All processes

Figure 4-11 Process Table view

5

Tuning database servers

The database server's primary function is to search and retrieve data from disk. Database engines currently available on Linux include IBM DB2® and Oracle. Database servers require high CPU power and an efficient disk subsystem because they issue many random I/O requests. A large amount of memory is also important, because it involves CPU subsystems.

A balanced system is especially important. When you add a new CPU, you should consider upgrading other subsystems as well, for example, by adding more memory and improving disk resources.

In database servers, the design of an application is critical (for example, database and index design).

This chapter covers the following topics:

- 5.1, "Important subsystems" on page 86
- 5.2, "Optimizing the disk subsystem" on page 86
- 5.3, "Optimizing DB2 memory usage" on page 88
- 5.4, "Optimizing Oracle memory usage" on page 90

5.1 Important subsystems

Key subsystems for databases are as follows:

- Processors

  CPU power is a important factor for database servers. Database queries and update operations require much CPU time. The database replication process also requires considerable numbers of CPU cycles.

  Database servers are multi-threaded applications, so SMP-capable systems provide better performance. Database applications scale very well, up to 16 processors and beyond. L2 cache size is also important because the hit ratio is high, approaching 90% in some cases.

- Memory

  The amount of memory is a very important performance factor for database servers. Buffer caches are the most important part of a database server, so they require large amounts of memory to maintain data in the system memory. If the server does not have sufficient memory, disks will be accessed, which generates latencies.

  Remember that system memory is also used for operating system needs; you should install enough memory for the OS to work properly, otherwise, paging is unavoidable.

  When compiling the kernel, make sure that the CONFIG_HIGHMEM64G_HIGHPTE option has been set to `y` when compiling a kernel. This enables all of the memory to be used by the database processes, and the page tables can grow and map to high memory.

- Disk

  Even with sufficient memory, most database servers will perform large amounts of disk I/O to bring data records into memory and flush modified data to disk. Therefore, it is important to configure sufficient numbers of disk drives to match the CPU processing power being used. In general, a minimum of 10 high-speed disk drives is required for each Xeon processor. Optimal configurations can require more than 50 10K RPM disk drives per Xeon CPU. With most database applications, more drives equals greater performance. For example, the xSeries 370 needs more than 450 10K RPM drives to reach maximum throughput when executing more than 40,000 transactions per minute.

5.2 Optimizing the disk subsystem

The main factors affecting performance include:

- The RAID controllers cache size
- The choice of RAID levels for the various data sets
- The RAID arrays stripe unit size
- The database block size

5.2.1 RAID controllers cache size

Depending on the storage access patterns, the RAID controller cache may have a major impact on system performance. The cache plays a particularly relevant role for write operations on RAID 5 arrays. Write buffering makes the RAID controller acknowledge the write operation before the write goes to disk.

This can positively affect performance in several ways:

- Updates can overwrite previous updates, thereby reducing the number of disk writes.
- By grouping several requests, the disk scheduler may achieve optimal performance.
- By grouping sequential write requests in the controller cache, small writes (operations updating a single stripe unit) may be converted into large writes (full stripe write operations updating all the stripe units of a stripe).

However, the relevance of cache size should not be overestimated. Whereas the existence of a small cache may deliver high benefits, often the marginal benefits of doubling the cache size are minimal. Moreover, it is important to remember that the RAID controller cache is only one of the many caches affecting I/O subsystem performance. A major role is played by the DBMS caches.

5.2.2 Optimal RAID level

Disregarding any cost constraint, the best choice is almost always RAID 10. The issue is to understand for which activities the delivered RAID 10 performance increase is worth the higher cost:

- In a data warehouse (DW), the typical access pattern to data files is almost 100% reading; therefore the better performance delivered by RAID 10 arrays in write operations is irrelevant and the choice of a RAID 5 level is acceptable.
- On the other hand, in a typical online transaction processing (OLTP) environment, the huge number of write operations makes RAID 10 the best level for data files arrays. Of course, even for the OLTP data files, RAID 5 may be acceptable in case of low concurrency (see below for an explanation of the concurrency concept).

For write operations, RAID 10 arrays are much better than RAID 5 arrays. However, for read operations, the difference between the two levels is minimal. From this simple rule stem the following recommendations:

- Online redo log files: RAID 1 is strongly recommended. In case of very high performance requirements, RAID 10 may be necessary. However, RAID 10 delivers performance benefits only in the case of quite small stripe unit size.
- Archive redo log files: RAID 1 is recommended. However, the archive redo logs are not as critical for performance as redo log files. Accordingly, it is better to have archive redo logs on RAID 5 arrays than to have redo logs on RAID 5 arrays.
- Temporary segments: RAID 1 or, even better, RAID 10, is recommended in case of many sort operations, as is typical, for instance, in data warehouses.
- Data files: RAID 5 is acceptable for data warehouses, because the typical access pattern is reading for small databases. Generally speaking, RAID 10 is the recommended RAID level.

5.2.3 Optimal stripe unit size

Typically, the stripe unit size should be a multiple of the database block size (for example, two times or three times the block size).

In addition, consider the average I/O size. Theoretically, the I/O size and the stripe unit size should be identical. However, because block boundaries are not necessarily aligned with stripe units, you should make the stripe unit size at least twice the average I/O size.

5.2.4 Database block size

The database block size is one parameter that significantly affects I/O performance. However, the only way to change this parameter once the database is created is to create a new DB and move data to it. The following basic rules of thumb may help in the decision making process:

- The typical database block size for OLTP systems is 8 KB or 16 KB.
- The typical DB block size for data warehouse (DW) systems is 16 KB or 32 KB, and perhaps even 64 KB.

In addition to this, the block size selected should be a multiple of the operating system basic block size (allocation unit), to avoid unnecessary I/O operations.

Oracle 9i supports multiple block sizes in the same database. This capability improves I/O performance because it is possible to select the best block size for each tablespace in a database.

5.3 Optimizing DB2 memory usage

How the Linux kernel manages physical memory is important to the performance of a database server. On an IA-32 architecture, the kernel manages memory in 4 KB pages. Most modern processors, however, are capable of working with larger pages (up to several megabytes).

For many applications, using 4 KB pages is suitable. Small pages reduce internal fragmentation, incur a smaller overhead when swapping data in and out of memory, and ensure that the virtual memory that is in use is resident in physical memory. Most database servers, however, use large shared memory segments for their database buffers. The result is that several page table entries (PTE) are required. This adds a considerable maintenance overhead for the memory manager.

Take, for example, a database server that allocates 1 GB of shared memory for its buffers using 4 MB shared memory segments. A kernel using 4 KB page size would require 262,144 PTEs, a significant number of page tables, which adds considerable maintenance overhead for the memory manager. Note, however, that this calculation is an oversimplification.The actual number of PTEs would be much larger because Linux is unable to share page tables for shared memory.

The use of large pages, on the other hand, can have a positive impact on database performance, especially for large database environments with many concurrent users. Large pages can reduce cache (table lookaside buffer, TLB) misses. Also, they reduce the number of page table entries (PTEs), which in turn reduces memory pressures on the system.

For example, suppose you have 1 GB of memory for buffer pools. The amount of memory required for the PTEs is 2 MB per agent. Assuming you have 500 concurrent users, this works out to a total of 1 GB memory used for PTEs, which can be a very big waste for a Linux system. The use of large pages will reduce the number of PTEs required.

5.3.1 Buffer pools

Proper memory utilization and data buffering is the key to database performance. All database activities at some point will require utilization of the various DB2 UDB buffer pools, caches, and heaps. The primary memory area is the buffer pool, which is the work area for DB2 data pages. A data page is the allocation unit where rows of table or index data are stored. The purpose of the buffer pool is to improve system performance. Data can be accessed much faster from memory than from disk; therefore, the fewer times the database manager needs to read from or write to a disk (I/O), the better the performance.

Key ideas for effective buffer pool utilization are:

- The larger the buffer pool, the more data can be stored.
- Keep more frequently accessed data in the buffer pool.
- Keep essential index data in the buffer pool for faster data access.

To accomplish these goals, changes to the size and number of buffer pools may be required. This can be accomplished manually or by using the DB2 UDB Performance Wizard.

DB2 lets you segment your data into different categories by creating separate table spaces for disparate data (frequently accessed data, history, index, sequentially accessed data, randomly accessed data, index data, LOB data). By segmenting the data, you can assign different buffer pools to corresponding table spaces, thereby controlling the data and system memory utilization.

A primary goal of performance tuning should be to minimize disk I/O. If I/O is necessary, it is important to make it as efficient as possible. There are two effective ideas for efficient I/O:

- Prefetching, the concept of moving data into the buffer pool before it is required by the application.

  When to do prefetching is largely a function of the database engine either determining it will be beneficial to prefetch data or, as a query is performed, detecting that prefetching will be helpful.

- Parallel I/O, moving data more quickly into the buffer pool by performing I/O operations simultaneously rather than sequentially.

  There are a number of ways to affect the amount of parallel I/O. The overall principle is to try to spread data access across as many physical drives as possible.

  RAID devices perform this task at a lower layer than the database engine. DB2 UDB can perform parallel I/O on a single RAID volume or across RAID volumes. If data is placed on a single RAID volume (for example, drive D:), the database engine does not know that the device is capable of performing multiple I/O operations simultaneously.

  The parameter DB2_PARALLEL_IO is used to inform the database engine that volumes are available for parallel I/O operations. To set this variable, open a DB2 command window and enter the command:

  ```
  DB2SET DB2_PARALLEL_IO=*
  ```

  This turns on parallel I/O for all volumes. As a general rule, this is a good idea when using hardware RAID devices. If database data is placed on multiple RAID volumes, it is automatically available for parallelism.

5.3.2 Table spaces

Data storage in DB2 UDB is based on the concept of table spaces. A table space is created from one or more containers. Containers are the locations for data placement and can be directories, specific files, or entire volumes.

There are two types of table spaces:

- System Managed Space (SMS)

  SMS table spaces are the simplest to administer. They contain one container, which is a directory where DB2 UDB will create and manipulate data files as needed and which is limited only by the size of the volume where the directory lives.

  This type of table space, however, cannot send table and index data pages into separate buffer pools. Also, data pages may not be contiguous because the OS has greater control over physical placement of data on the volume.

- Database Managed Space (DMS)

  DMS table spaces have greater control over data placement. The containers for a DMS table space are either files of a specified size or entire raw volumes. If using file containers, there should only be one file per volume and each container should be of the same size. As DMS table space containers fill up, you may either increase the size of the containers if the containing volumes have available space, or you may add containers to the table space.

  The DMS table space type allows for table and index data to be separated into different table spaces, thereby separating buffer pools. DMS data is also more likely to be stored contiguously, making for more efficient I/O.

The database configuration parameter NUM_IOSERVERS specifies how many database agents are available for performing prefetching and parallel I/O operations. This should be set to one or two more than the number of physical drives that make up the volumes where DB2 data is stored.

Another important I/O operation is logging. Because all database data changes must be logged in order to guarantee data consistency, it is important that the logging activity does not become a bottleneck. The DB2 UDB database logs should be placed on a volume with enough physical drives to meet the write-intensive work of the logger. The database configuration parameter NEWLOGPATH is used to specify the path where the database logs are created.

5.4 Optimizing Oracle memory usage

This section describes tuning activities related to Oracle and the use of memory.

Oracle instance memory tuning is one of the areas where small parameter changes can produce a big increase, as well as a big decrease, of performance. In this section, we describe the shared pool, database buffer tuning, and redo log buffer.

5.4.1 Shared pool

The shared pool is composed of two areas: the library cache and the dictionary cache.

- Library cache

  The library cache includes three memory areas:

  - Shared SQL area: Contains the execution plans of parsed SQL statements.
  - PL/SQL programs: Contains PL/SQL programs in compiled forms.
  - Locks

Oracle dynamically tunes the relative size of these areas. The only manually tunable parameter is the shared pool global size variable SHARED_POOL_SIZE. To see whether the library cache is properly sized, the following simple SQL instruction can be used:

```
select round((sum(pins-reloads)/sum(pins))*100,2) as hit_ratio
from v$librarycache;
```

The term `pins` refers to the number of times a parsed SQL statement was looked for in the library cache. The `reloads` are the number of times the search was unsuccessful. The library cache hit ratio should be as high as 99%. If the hit ratio is lower, either the instance has been recently started, so the cache is sub-optimal, or the shared pool is insufficient and the size should be made larger.

- Dictionary cache

  this simple SQL instruction shows whether the size of the dictionary cache is optimal:

  ```
  select round((sum(gets-getmisses)/sum(gets))*100,2) as hit_ratio
  from v$rowcache;
  ```

  The term `gets` refers to the number of times a request was made for information in the dictionary cache, while the term `getmisses` refers to the number of unsuccessful requests. The dictionary cache hit ratio should be as high as 99%. If the hit ratio is lower, either the instance has been recently started, so the cache is sub-optimal, or the shared pool is insufficient and the size should be made larger.

5.4.2 Database buffer cache

Server foreground processes read from data files into the database buffer cache, so the next readings need no I/O operation. Server processes also write modified data into the database buffer cache. Asynchronously, a dedicated background process (DBW*n*) will move dirty data from the cache to the data files. In this way, I/O performance is greatly increased. The performance benefits obviously depend on cache hits, that is, on how many times server processes looking for data find them in the cache. The following section describes the internal structure of the buffer cache and how to tune it.

Buffer cache architecture

The buffer cache is composed of as many buffers as the value of the init<sid>.ora parameter DB_BLOCK_BUFFERS. The sizes of the buffers are identical and correspond to the init<sid>.ora parameter DB_BLOCK_SIZE. The buffer cache is filled in by foreground processes reading data from data files and flushed out by the DBW*n* process when one of the following events occurs:

- DBW*n* time-out (each three seconds)
- Checkpoint
- No free buffer

Data is removed from the buffer cache according to the least recently used algorithm. Moreover, in order to avoid cache quality worsening due to single full table scan instructions, Table Access Full operations are always put at the end of LRU lists.

Optimal buffer cache

To see whether the size of the buffer cache is optimal, the following query can be used:

```
select name, value
from v$sysstat
where name in (‘db block gets’, ‘consistent gets’, ‘physical reads’);
```

Given the output of this **select** command, the buffer cache hit ratio can be obtained with the following simple calculation:

$$\frac{dbblockgets + consistentgets - physicalreads}{(dbblockgets + consistentgets)} \times 100$$

Enlarging the buffer cache

The buffer cache hit-ratio should be 90% or higher. Values between 70% and 90% are acceptable in case it is necessary to resize the buffer cache to improve the library or dictionary hit cache ratios. If the buffer cache hit ratio is too low, the optimal number of buffers to add to the cache can be obtained with the following complex query on the V$DB_CACHE_ADVICE view:

```
column size_for_estimate format 999,999,999,999 heading 'Cache Size (m)'
column buffers_for_estimate format 999,999,999 heading 'Buffers'
column estd_physical_read_factor format 999.90 heading 'Estd Phys|Read Factor'
column estd_physical_reads format 999,999,999 heading 'Estd Phys| Reads'

SELECT size_for_estimate, buffers_for_estimate, estd_physical_read_factor,
estd_physical_reads
FROM V$DB_CACHE_ADVICE
WHERE name = 'DEFAULT'
AND block_size = (SELECT value FROM V$PARAMETER
   WHERE name = 'db_block_size')
   AND advice_status = 'ON';
```

Before running the query, the V$DB_CACHE_ADVICE view has to be activated by using the following command:

```
alter system set DB_CACHE_ADVICE=ON;
```

Moreover, in order to obtain significant results, a representative workload should have been running on the system for a reasonable time interval, thereby achieving a stable buffer population.

Because the activation of the V$DB_CACHE_ADVICE view has a (minor) impact on CPU load and memory allocation, at the end of the analysis, it is recommended to de-activate the view with the command

```
alter system set DB_CACHE_ADVICE=OFF;
```

The output of the query is a set of lines showing the incremental benefits of the various cache sizes. The first column of the query output contains the various cache sizes while the latter column shows the physical reads. Upon increasing the cache size, the physical reads decrease, but the incremental benefits also decrease.

Multiple buffer pools

Starting with Oracle 8, multiple buffer pools can be created and separately sized. The database buffer cache is composed of the following three buffer pools:

- Keep pool
- Recycle pool
- Default pool

The keep pool stores data that must not be moved out of the buffer cache. The recycle pool is for data that must be quickly moved out of the buffer cache when no longer necessary. Everything else is in the default pool. Unlike the shared pool, whose internal memory areas (library cache, dictionary cache) cannot be separately sized, it is possible to size the keep pool and the recycle pool, and also, as a result, the default pool.

The following example shows how to size a 1000-buffers buffer cache so that 50% is used for the recycle pool, 25% for the keep pool, and 25% for the default pool:

```
DB_BLOCK_BUFFERS=1000
DB_BLOCK_LRU_LATCHES=20
BUFFER_POOL_RECYCLE=(buffers:500, lru_latches:10)
BUFFER_POOL_KEEP=(buffers:250, lru_latches:5)
```

Latches are memory locks and should be sized according to the following rule: 1 latch for each 50 buffers, as in the example above.

5.4.3 Redo log buffer cache

Each server process that updates data first needs to update the redo log files. To improve performance, server processes only write redo log entries into the redo log buffer cache, while the LGWR process is responsible for moving dirty buffers from memory to disks. To avoid buffer data corruption, a locking mechanism (latch) is used, so that only one process at a time can write on the redo log buffer cache. Given the sequential nature of redo log data, only one redo log allocation latch is made available to Oracle server processes. As a result, redo log buffers can be a source of delay because of high resource contention.

To see whether there is an excessive redo log buffer contention, the following query can be used:

```
select name, value
from v$sysstat
where name='redo buffer allocation retries';.
```

Any value other than 0 shows that processes had to wait for space in the redo log buffer cache.

The size of the redo log buffer can be configured by changing the LOG_BUFFER parameter in the init<sid>.ora file. This parameter gives the value in bytes of the cache and must be a multiple of DB_BLOCK_SIZE.

Each server process wanting to write to the redo log buffer cache must first get the redo allocation latch. The process will then write as many bytes as allowed by the LOG_SMALL_ENTRY_MAX_SIZE parameter in init<sid>.ora. When this number of bytes has been written, the process must release the latch in order to allow other processes to have a chance of acquiring the lock. To increase the ability of server processes to work concurrently, it is recommended that you size the LOG_SMALL_ENTRY_MAX_SIZE parameter as small as possible.

6

Tuning Samba for file and print

File sharing and printing are an important part of the modern office computing environment. In this chapter, we introduce Samba, which is a popular file and print services application that enables Linux to provide these services to Windows SMB/CIFS-based clients. Samba is powerful Open Source GNU Licensed software.

The topics covered in this chapter are as follows:

- 6.1, “Important subsystems” on page 96
- 6.2, “Sizing and optimizing the hardware” on page 96
- 6.3, “Samba specific optimizations” on page 97

Tip: The concepts in this chapter focus on Samba but can be applied to most file sharing applications.

6.1 Important subsystems

A Samba server acts as a file and print server for SMB/CIFS clients. The important subsystems for Samba are:

- Network
- Disk
- Memory

Refer to Chapter 3, “Analyzing performance bottlenecks” on page 51 for more information about detecting and removing bottlenecks related to these subsystems.

As you identify which subsystems are potential sources of bottlenecks, you will have a general idea of what initial actions to take regarding performance optimization. It should be noted that optimizing file sharing involves the client, and not all bottlenecks are caused by the server. Care should be taken to ensure that the clients have been tuned appropriately so that they can get maximum networking performance.

6.2 Sizing and optimizing the hardware

This section describes different aspects of the key hardware subsystems.

6.2.1 Network

The main purpose of any file and printer sharing is to provide resources across the network, so the network plays an important role. The rules for sizing and determining the network requirements on Samba can be applied to other file sharing protocols/applications. An easy formula to calculate the estimated bandwidth required is:

number of clients x transfers per second x average file size (KB) = KBps used

So, for example, if we have an office of 500 users and each user performs two transactions a second on a file that is 25 KB, our bandwidth used would be approximately 25 MBps.

6.2.2 Disk

Because the major function of the file server involves disk I/O, serious consideration should be given to this area of the server.

We recommend the following disk configuration:

- Use RAID-1E, RAID-10, RAID-5/5E, or RAID-50 for hard drive fault tolerance. RAID-1E and RAID-10 will give you better performance, but RAID-5 will give you more usable space for the same number of disks.
- Place the operating system code, swap partition and Samba application code on a drive (or drives) separate from the shared data. The operating system code should not be accessed very often after the server has booted, and provided the server has enough memory, the swap partition should not be accessed very often either.
- For ServeRAID-based systems, enable write-back cache (with the battery-backed cache, this is the default; without it, the default is write-through). This enables all writes to disk to be stored in a cache, which permits the system to continue operations without waiting for confirmation that data has been written to the hard disk.

- Install the latest disk controller BIOS, firmware, and drivers (for instance, ServeRAID or Fibre Channel).
- Use the fastest drives possible; for more information, see Table 3-2 on page 59.

6.2.3 Memory

For each user who connects to a Samba server, a new smbd daemon is spawned and uses, on average, approximately 1 MB to 2 MB of memory. Based on this, one can roughly calculate how much memory will be used by the Samba application.

6.3 Samba specific optimizations

The parameters for tuning Samba are located within the Samba configuration file /etc/samba/smb.conf.

Socket options are targeted specifically at the TCP networking layer. Within Samba, these options can be specified on the command with the **-o** option or with the **sock options** parameter within the Samba configuration file. SUSE LINUX Enterprise Server 8 has these already set for good performance, but ensure that the following options are set for a local area network install of Samba:

```
socket options = IPTOS_LOWDELAY TCP_NODELAY
```

If you are installing Samba into a wide area network, try the following options:

```
socket options = IPTOS_THROUGHPUT TCP_NODELAY
```

For more information about the available socket options parameters, consult the smb.conf man page.

The log level option can also dramatically decrease performance. If this option is set to a value higher than 2, a production machine will suffer from degraded performance due to excessive logging.

In a RAID environment, you will want to set the write cache size to be equal to the RAID stripe size; the write cache parameter is used to control when certain information is saved to disk. For example, if you have a stripe size of 32 K, you would set the parameter to 32768 because Samba reads the value in bytes.

```
write cache size = 32768
```

Other options that may improve performance but should be tested in your environment are:

```
read raw = no
write raw = no
```

7

Tuning LDAP

Lightweight Directory Access Protocol (LDAP) is a technology for accessing common directory information. LDAP has been embraced and implemented in most network-oriented middleware. As an open, vendor-neutral standard, LDAP provides an extendable architecture for centralized storage and management of information that must be available for today's distributed systems and services. LDAP has become the de facto access method for directory information, much as DNS is used for IP address look-up on the Internet.

For example, LDAP can be used to centralize login information for all the systems on your network. This replaces the need to maintain that information on each system, normally stored in /etc/passwd and /etc/shadow. An LDAP object for a user may contain this information:

```
dn: uid=erica,ou=staff,dc=redpaper,dc=com
cn: Erica Santim
uid: esantim
uidNumber: 1001
gidNumber: 100
homeDirectory: /export/home/esantim
loginShell: /bin/ksh
objectClass: top
objectClass: posixAccount
```

The objects above are identified by the DN (name) attribute that works like a primary key in any database.

This chapter includes the following topics:

- 7.1, “Hardware subsystems” on page 100
- 7.2, “Operating system optimizations” on page 100
- 7.3, “OpenLDAP 2 optimizations” on page 100

For details about LDAP implementation and optimization, see the IBM Redbook *Understanding LDAP - Design and Implementation*, SG24-4986.

7.1 Hardware subsystems

An LDAP directory is often described as a database, but it is a specialized database that has characteristics that set it apart from general purpose relational databases. One special characteristic of directories is that they are accessed (read or searched) much more often than they are updated (written). Hundreds of people might look up an individual's phone number, or thousands of print clients might look up the characteristics of a particular printer. But the phone number or printer characteristics rarely change. LDAP directories are typically optimized for read access.

Key hardware subsystems are:

- Swap and physical memory
- CPU utilization and I/O in disks

Consider the following recommendations:

- Memory is often the bottleneck for LDAP servers. Ensure thatyour server has enough installed.
- Separate the LDAP directories and the logs onto separate disks or RAID arrays.
- The LDAP server daemon, slapd, uses memory and CPU to perform indexing in order to improve look-up times. Cache usage is controlled by settings in slapd.conf.

7.2 Operating system optimizations

We can manage our server's performance by limiting the amount of resources the server uses to process client search requests. The following are parameters that can be altered. Ongoing monitoring of the system is important.

- The time during which the server maintains the FIN-WAIT-2 connections before terminating them is controlled by tcp_fin_timeout:

  ```
  sysctl -w net.ipv4.tcp_fin_timeout=15
  ```

- The maximum number of file descriptors that Linux can have open at any one time is specified by file-max:

  ```
  sysctl -w fs.file-max=131063
  ```

- Disable the writing of access timestamps to files to increase I/O performance by adding the noatime parameter to /etc/fstab as follows:

  ```
  /dev/sda2 / reiserfs defaults,noatime 1 1
  ```

- Enable asynchronous I/O so that a process does not have to wait for the I/O response to be ready:

  ```
  /dev/sda2 / reiserfs defaults,noatime,async 1 1
  ```

7.3 OpenLDAP 2 optimizations

Here are some tuning parameters that may improve performance of your OpenLDAP installation. Make these changes in the file /etc/openldap/slapd.conf:

- idletimeout specifies how long the LDAP server should wait (in seconds) before closing an idle client connection. The default is 0 (disable), but two minutes is enough:

  ```
  idletimeout 120
  ```

Tip: Don't use idletimeout together with the `tcp_tw_reuse` kernel parameter because LDAP drops reused connections. If you need to solve problems with excessive TCP connections, we recommend you instead disable idletimeout and configure tcp_tw_reuse and tcp_tw_recycle.

- Limit the maximum number of entries that can be returned from a search operation. This is especially important when clients uses wild cards on searches. The default is 500.

  ```
  sizelimit 100
  ```

- Specify the maximum number, in seconds, that OpenLDAP spends answering a client request. The default is 3600 (60 minutes). Normally, you can reduce this to 10 minutes.

  ```
  timelimit 360
  ```

- Disable all logging unless you need the information:

  ```
  loglevel 0
  ```

7.3.1 Indexing objects

Indexing LDAP objects ensures that searches on those objects are quicker and consume less server resources. However, indexing objects that are not used or rarely used for searches also impacts performance.

OpenLDAP enables you to specify what objects are indexed using the index statements in the slapd.conf file:

```
index {<attrlist> | default} [eq,sub,pres,approx,none]
```

attrlist specifies which attributes to index. The second parameter specifies what type of index is created:

- `eq` means equality: an exact match. For example, `cn=optical` only returns results where cn is exactly the string `optical`. `eq` implies `pres`, so adding `pres` is unnecessary.
- `sub` means substring: for example, `cn=*optical*` returns all values of cn containing the string `optical`.
- `pres` means presence: if the attribute is present in an entry.
- `approx` means approximate: where the value sounds like the search result. This is based on the enable-phonetic compile parameter.
- `none` means Not indexed.

For example:

```
index cn
index sn,uid eq,sub
index default none
```

The first line specifies that the LDAP will index attribute cn for all kinds of searches available. The second line specifies that attributes sn and uid will only be indexed when the search includes eq and sub. The third line specifies that no others attributes will be indexed.

By default, no indices are maintained. It is generally advised that at a minimum, an equality index of objectClass be maintained.

7.3.2 Caching

Frequent searches will benefit from the use of cache. This portion of memory will store the data used to determine the results, thereby making common searches much faster.

To specify the amount of memory used for the cache associated with each open index file, use the dbcachesize parameter, in bytes. The default is 100 000:

```
dbcachesize 100000
```

You can also specify how many attributes of the most frequent searches OpenLDAP will put in memory. The default is 1000 entries:

```
cachesize 1000
```

8

Tuning Lotus Domino

Lotus® Domino® is a popular application and messaging system that enables a broad range of secure, interactive business solutions for the Internet and intranets. The Lotus Domino server is a powerful tool for organizational communication, collaboration, and information sharing.

Just as for other application servers, careful planning, maintenance and tuning processes are essential in systems administration. This chapter discusses the tuning concepts and procedures of Lotus Domino R6.5 server running on Linux.

Topics covered are:

- 8.1, “Important subsystems” on page 104
- 8.2, “Optimizing the operating system” on page 106
- 8.3, “Domino tuning” on page 107

The chapter is based, in part, on the Redpaper *Domino for IBM @server xSeries and BladeCenter Sizing and Performance Tuning*, REDP3851, which is available from:

http://www.redbooks.ibm.com/abstracts/redp3851.html

8.1 Important subsystems

Lotus Domino server can act as a mail, Web, application, hub, and database server handling mail routing and storage as well as database and Web requests. Because of the dynamic nature of Lotus Domino, the important subsystems that can be sources of a bottleneck are:

- Network
- Memory
- CPU
- Disk

Refer to Chapter 3, “Analyzing performance bottlenecks” on page 51 for more information about detecting and removing bottlenecks related to these subsystems.

As you identify which subsystems are potential sources of bottlenecks, you will have a general idea of what initial actions to take regarding performance optimization.

8.1.1 Network adapter card

When analyzing network performance and potential bottlenecks, the following parameters should be investigated to maximize performance:

- Network adapters
- Ethernet adapter performance
- TCP/IP performance
- Operating system settings
- Network design

Network compression is an important performance feature offered in Domino 6.5. When you enable network compression, data is automatically compressed before it is sent over the network. This improves network performance, especially over slower line speeds. Network compression can result in 34% to 52% less network traffic and enable faster message delivery.

> **Tip**: Never assume that you do not have a LAN bottleneck simply by looking at LAN sustained throughput in bytes/sec. LAN adapter bottlenecks often occur at low LAN utilization but high sustained packet rates. Observing packets/sec will often yield clues as to these types of bottlenecks.

We recommend the following network adapter configuration for your server, especially if your Domino system is utilized by a large number of users:

- Use the onboard Ethernet adapter (10/100/1000 Mbps) in the server where possible.
- A Gigabit Ethernet connection from the server to the Ethernet switch is recommended.
- Use an IBM PCI-X Ethernet adapter with full-duplex capability.
- For servers that will support a large number of users, consider using multiple network adapters.

8.1.2 Server memory

The minimum amount of random access memory (RAM) you need in your Domino server depends on two primary factors. Domino and the OS need a base amount of RAM to be able to execute on the hardware; this is the first component of the total RAM required. The second component depends on how many Notes and Web clients have sessions with your server and on what other tasks your server might have to perform. The memory model has been redesigned in Domino 6.5, and this has meant a 30% reduction in memory requirements per user.

Use the following formula to calculate the minimum amount of RAM you need in your Domino 6.5 server.

Minimum memory requirements: 128 MB + (number of concurrent users/5) MB

Remember that this RAM requirement includes only the OS and Domino running on your server. If you have any file and print services, backup and recovery, or anti-virus software running on your server, you will need to configure additional RAM for your server. If the above calculation determines you need 158 MB of RAM, for example, install 256 MB.

This algorithm is appropriate for mail and application servers and mail hubs. However, it is not appropriate for replication hubs. Usually, replication hubs are heavily used but have few, if any, sessions open to active users.

Note: Lotus Domino only addresses up to 2 GB of RAM. If you install extra memory, it will only be used by the operating system and other additional tasks. This means that installing more than 2.5 GB of RAM will not significantly improve server performance.

Total memory minus available memory equals the amount of memory the server is actually using, sometimes called the “working set.” Because memory use is dynamic, it is best to monitor memory utilization over an extended period of time to arrive at an accurate representation of the memory working set. To determine the amount of memory needed to support double the users, double the working set size, and then add 30% as a buffer for peak activity.

Servers should be configured so that average memory utilization does not exceed 70%. If this level is consistently exceeded, you run the risk that the server will expand storage onto disk (page memory onto disk) during periods of peak activity. Servers should *never* regularly page memory to disk, because performance will be very adversely affected.

Memory guideline: Paging I/O should occur only occasionally, such as when applications are initializing, never on a continuous basis. Average memory utilization should not exceed 70% of available memory.

8.1.3 Processors

The CPU requirement per user has been reduced by as much as 23% with Domino 6.5, enabling servers to handle more users.

Many Domino functions are processor-intensive. Tasks such as routing messages, indexing databases, searching databases, and dynamically creating HTML pages for Web clients all put heavy demands on the system’s CPU. We therefore recommend that you purchase an xSeries server that supports at least two-way SMP. You might not need or be able to afford the extra CPUs today, but you are providing room to grow as your use of Domino functions expands.

Domino can take advantage of the benefits of SMP. It does so automatically; you do not need to tune or configure your hardware for Domino to take advantage of the additional CPUs. Domino takes advantage of multiple processors by allocating the replicator and router tasks to different CPUs, thereby spreading the load.

8.1.4 Disk controllers

Several xSeries servers have redundant array of independent disks (RAID) technology as their standard. This allows data to be striped across a number of hard disks that are seen by the OS as one large disk drive.

We recommend the following for your Domino server:

- Implement RAID-1E or RAID-10 on all of your servers if financially possible, or RAID-5 or RAID-50 as your second choice. As your Domino servers become more critical to your business, you will want the best combination of disk fault tolerance and performance you can afford.
- Implement a second RAID-1E array to store the Domino 6 transaction logs.
- Do not implement multiple logical drives or partitions on the same RAID array. For each RAID array, create only one logical drive, and on that logical drive, only one partition. This will reduce latency due to extreme drive head movement.
- Only use hardware-controlled RAID arrays. Software-controlled RAID arrays place additional strain on your server in terms of CPU utilization. The CPU should be dedicated to providing Domino functions, not implementing a RAID subsystem.
- If possible, install an additional hard disk for use as a hot spare. Hot spares can replace failed disks automatically, or can be managed by the administrator.
- Choose the fastest hard drives. High-performance disks provide fast data rates and low latency times. xSeries servers support very fast 15,000 RPM drives.
- Do not mix SCSI Fast with SCSI Fast/Wide drives.
- Do not mix drives that spin at different speeds.
- Do not mix drives of different sizes in a RAID array.
- Enable write-back cache on the RAID controller. This setting improves system performance by caching information before it is written back to disk. Some controllers do not have battery backups, and cached data will be lost in the event of a power failure. In this situation, installing an uninterruptible power supply (UPS) for your server is particularly important.
- Set the RAID stripe size to 16 KB.
- Install the latest BIOS, firmware, and drivers for the disk controller (ServeRAID or Fibre Channel, for example).

8.2 Optimizing the operating system

There are a number of kernel parameters that can affect Domino server performance. For more information about tuning these parameters, refer to Chapter 1, “Tuning the operating system” on page 1.

- fs.file-max

  This specifies the maximum number of file handles that can be opened by Domino. The command **`sysctl fs.file-max`** shows the current value.

  Ensure the value is at least 49152 (48 x 1024). You may find the default value to be higher than this value; in that case, leave it unchanged.

  ```
  sysctl -w fs.file-max=49152
  ```

- kernel.shmmni

  This specifies the maximum number of shared memory segments for the OS. Ensure the value is at least 8192 by checking its value using **`sysctl kernel.shmmni`**. If it is smaller, set it to `8192` with the command:

  ```
  sysctl -w kernel.shmmni=8192
  ```

- kernel.threads-max

  This specifies the maximum number of threads for the OS. Ensure the value is at least 8192 by checking its value using **`sysctl kernel.threads-max`**. If it is smaller, set it to `8192` with the command:

  ```
  sysctl -w kernel.threads-max=8192
  ```

- Set **`noatime`** on the file systems containing the Domino data directories, as described in "Access time updates" on page 20.

8.3 Domino tuning

As discussed, Lotus Domino servers can act as a mail, Web, application, hub and database server. It may need to handle mail routing, storage, and database and Web requests. Parameters within Domino 6.5 can be adjusted to assist and maximize the performance of the server based on the applications that it will run.

It might also be necessary to modify server settings to manage customized development requirements (indexing, APIs, and so on).

This chapter provides Domino tuning practices that will assist you in gaining greater server performance.

The best way to improve Domino performance is to:

1. Plan ahead.
2. Test the Domino application thoroughly on a test system and then on a pilot system.
3. Start performance monitoring from day one, before the first user signs on.
4. Constantly analyze your performance data and adjust your sizing based on actual facts.
5. Roll it out gradually, a few users at a time, constantly monitoring its performance.
6. Continue to analyze the performance data and report future trends and hardware needs.

Lotus Domino installed straight out of the box is optimized for most configurations. However, there are some tuning parameters in specialized situations that can improve the performance of your Lotus Domino server and also protect the server from overloading.

8.3.1 The notes.ini file

The Lotus Notes® Domino server is controlled by the values stored within the notes.ini file. By default, this file is located in /local/notesdata/.

> **Important:** It is strongly recommend that you take a back-up of your notes.ini file before you make any modifications to it.

Configuring server tasks

Each task increases the server's load and can adversely affect its performance. Minimizing the number of server tasks that are run by the server, the frequency at which they run, and the time in which they run will enable you to increase the performance of the server.

Example 8-1 notes.ini servertasks list

```
ServerTasks=Update,Replica,Router,AMgr,AdminP,CalConn,Sched
ServerTasksAt1=Catalog,Design
ServerTasksAt2=UpdAll
ServerTasksAt3=Object Info -Full
ServerTasksAt5=Statlog
```

Each of these variables controls the schedule for automatic server and database maintenance tasks. In Example 8-1, the first `ServerTasks` line denotes the services that are run when the Domino task starts while the other lines denote the scheduled server tasks. The time is entered in a 24-hour format, where 0 is 12:00 a.m. and 23 is 11:00 p.m. In the example above, Catalog and Design tasks would start at 1:00 a.m., and the Statlog task would start at 5:00 a.m.

You can significantly improve performance by removing tasks that are not appropriate to the server. Consider the following suggestions to increase performance related to Lotus Domino server tasks:

- These tasks should be turned off when they are not in use:
  - Scheduling: Turn off this task if you are not using the server for scheduling and calendaring.
  - AMgr: Turn off this task if you are not using the server to run scheduled agents. Remember, this function is not required for WebQuery Agents.
  - Collector, Reporter: Turn off this task if you are not using the server to automatically track server statistics on a regular basis. You can collect them on demand.
- Remove the Replicator (Replica) and Router tasks.

  Both of these tasks can be removed if they are not being used on the server, because each takes up a fair amount of server resources when loaded. For example, if you have only one Lotus Domino server in your organization that is used for both applications and mail routing, you might not need the Replicator task, because you do not have any other servers from which to replicate (and because clients will be replicating from the Lotus Domino server, not vice versa).
- Carefully choose the times when server tasks are run.

  Daily server tasks should not be run when other server tasks are running or at times when there are many users using the Lotus Domino server. This allows the maximum amount of server resources to be available for each server task that is currently executing and for user sessions. Examples of such server tasks are Design, Catalog, Statlog, and customized Notes API programs that need to be run only once a day.

The entries in ServerTasks have the following uses:

- ADMINP: The major Domino scheduler job running batch-like updates
- AMGR: Domino Agent Manager; takes care of predefined agents
- CALCONN: Connects the Domino calendar to other types of calendars
- EVENT: Domino Event Handler
- REPLICA: Domino database Replicator task
- REPORT: Domino Report Generator (adds data to STATREP.NSF)
- ROUTER: Domino Mail Routing task

- SCHED: Domino Scheduler task
- SERVER: The main Domino Server task
- UPDATE: Updates the database indexes and views

> **Tip:** A very obvious but important lesson about server tasks and functions is that if your organizational requirements have no need of a particular function, you should not run it on your server!

Domino database indexing: controlling the Update task

The Update task is designed to run in the background and is intended to improve response time and performance by ensuring that when a user opens a database view, he or she does not have to wait for it to be indexed.

> **Note:** Do not remove the Update task from a server. If you do so, the Public Address Book will not be updated.

To improve view-indexing performance, you can run multiple Update tasks if your server has adequate CPU power. Doing this can affect server performance and is recommended primarily for multiprocessor machines. On a server with multiple processors, enable a maximum of one Update task per processor.

This is done within the notes.ini file by adding the line:

```
Updaters = [number of processors]
```

Network performance (compression)

Network compression is an important performance feature offered in Lotus Notes/Domino 6. When you enable network compression, data is automatically compressed before it is sent over the network. This improves network performance, especially over slower lines.

Notes/Domino 6 network compression offers a number of immediate benefits. For example, by reducing the amount of data being transmitted, you can improve the performance of your routing and replicating hub servers, especially if they are currently laboring under heavy workloads. In addition, you can enable network compression by default, so all your users can take advantage of this functionality without having to select it themselves. Because network compression is a standard, out-of-the-box feature, it does not require any additional code, which helps simplify administration and requires fewer CPU resources to run.

Note the following statistics about the benefits of network compression:

- A 35-52% reduction in data transferred from server to client
- A 26% reduction in data transferred from server to server

Modify the TCP/IP line in the notes.ini file to enable network compression. The last parameter denotes compression:

```
TCPIP = TCP,0,15,0,,12320
```

Setting maximum mail threads for local delivery

The MailMaxDeliveryThreads setting determines the maximum number of threads the router can create to perform local mail delivery. The default number is 1. Increasing this value can improve message throughput for local deliveries. The ideal number usually falls between 3 to 25, depending on the number of local deliveries on your Lotus Domino mail server.

```
MailMaxDeliveryThreads = [number]
```

Disabling per-user message caching by the IMAP task

This setting disables per-user message caching by the IMAP task. It can improve the capacity of a server by reducing memory consumption. However, the response time for some user operations might be slower. If this setting is omitted, IMAP per-user message caching will be enabled.

To disable per-user message caching by the IMAP task, set the following value to 1:

```
NoMsgCache = [0 or 1]
```

Using the Lotus Domino 6.5 database format

Databases that you create in Lotus Domino 6.5 perform considerably better than databases created in previous releases. Database operations require less I/O and fewer CPU resources, view rebuilding and updating are quicker, and memory and disk space allocation is improved.

Because of these performance improvements, limiting the size of the databases to improve database performance is less important than in past releases. The maximum database size for Lotus Domino 6.5 format databases is 64 GB.

Defining the number of databases cached simultaneously

If your server has sufficient memory, you can improve its performance by increasing the number of databases that Lotus Domino can cache in memory at one time. To do so, use the NSF_DbCache_Maxentries statement in the NOTES.INI file. The default value is 25 or the NSF_Buffer_Pool_Size divided by 300 KB, whichever value is greater. The maximum number of databases that can be cached in memory is approximately 10,000. For short intervals, Lotus Domino will store up to one and a half times the number entered for this variable.

Monitor the Database.DbCache.Hits statistic on your server. This indicates the number of times a database open request was satisfied by finding the database in cache. A high value indicates that the database cache is working effectively. If the ratio of Database.DbCache.Hits to InitialDbOpen is low, you might consider increasing NSF_DbCache_Maxentries.

To set the number of databases that a server can hold in its database cache at one time, set the NOTES.INI value as follows:

```
NSF_DbCache_Maxentries = [number]
```

In special circumstances, you might also want to disable the database caching. The database cache is enabled by default. To disable database caching, enter the following syntax on the Domino console:

```
Dbcache disable
```

The database cache keeps databases open. Use this command to disable the database cache when you need exclusive access to a file that might be in it. For example, to run an application such as a virus checker or backup software, disable the cache. To re-enable the database cache, restart the server.

Optimizing database index update

In general, the fewer view indexes that the Indexer server task must update, the fewer server resources are used by this task. You can use the NOTES.INI variable Default_Index_Lifetime_Days to minimize the amount of effort required by the Indexer task when updating view indexes. The Default_Index_Lifetime_Days variable controls how long a view index is kept in a database before it is deleted due to non-use:

```
Default_Index_Lifetime_Days = [number of days]
```

Full-text indexes

Disable the updating of full-text indexes on a server if you do not have any full-text indexed databases on your server (and do not intend to have any). The NOTES.INI variable Update_No_Fulltext can be used to disable all full-text index updating on the server. You might want to use this variable to disable the updating of full-text indexes if, for example, you do not want users to create full-text indexes of their mail files on a mail server in order to save disk space on that server and save the Indexer task the time and resources needed to update these full-text indexes. This is a very good setting for mail and replication hub servers, which in most circumstances do not have any user connections. The full-text index variable is:

```
Update_No_Fulltext = [0 or 1]
```

Setting this value to 0 causes full-text indexes to be updated each time the Update task (a server command task) is executed, and setting the value to 1 disables all full-text indexing.

Maximum sessions

When a new user attempts to log on, if the current number of sessions is greater than the value of Server_MaxSessions (in the NOTES.INI file), the Lotus Domino server closes the least recently used session. In order for a session to be considered for closing, it must have been inactive for at least one minute.For example, if this parameter is set to 100, and the 101st person tries to access the Lotus Domino server, the Lotus Domino server drops the least-used session from the server in favor of this new session.

Reducing maximum server sessions: Reducing the Server_MaxSessions value to a specific number will not prevent the server from allowing more than that number of concurrent active users on the server, but will drop the sessions soon after they become inactive. This frees up resources. Conversely, Domino will not close any session that has been idle for less than one minute regardless of the demand on the server.

Controlling minimum mail poll time

You can control the minimum allowable time in which Lotus Notes clients can poll for new mail. It is possible that your Lotus Domino server resources are being overtaxed by being constantly bombarded by requests for new mail from the Notes client machines if users have changed their default new mail notification check time from 15 minutes to a smaller number such as 2 or 5.

You can control the minimum frequency of these requests from the server by using the MinNewMailPoll NOTES.INI variable. This variable determines the minimum allowable checking time from clients regardless of the value specified on the client machines. No default is set during server setup. The syntax of this variable is as follows:

```
MinNewMailPoll = [minutes]
```

Setting up multiple replication tasks

You can improve server replication performance by running multiple replicator tasks simultaneously. By default, only one replicator task is executed. With a single replicator task running, if you want to replicate with multiple servers at the same time, the first replication must complete before the next one can start. Set the number of replicators by adding the following entry in the NOTES.INI file, where the value is the number of replicators:

```
Replicators = [number]
```

All other factors aside, it is recommended that you set the number of replicator tasks equal to the number of spoke servers with which the hub replicates. However, you should not exceed 20 replicators to avoid putting too much load on the server. If the server you intended to replicate is not a hub server, the recommended number of replicators should equal the number of processors on the server.

8.3.2 Enabling transaction logging

Transaction logging is available in Lotus Domino 6.5 servers. With transaction logging, each transaction is transferred to its database, not directly, but by posting sufficient information to complete the transaction to a high-speed access sequential log file. The transaction is finally posted to the database at some later time. Data in the log, but not yet posted to the database, can be recovered in the event of a server failure.

Tests have shown that enabling transactional logging provides a performance advantage of up to 20%, depending on the exact server configuration. The Lotus Domino transaction log is actually a set of sequential access files used to store the changes made to the databases. Sequential disk access to a dedicated physical disk is noticeably faster than random disk access.

Transaction logging disk optimization: For optimal performance, transaction logs should be placed on a separate physical disk device. If you place the logs on the same device as the databases, you lose the benefit of sequential access, and there is no performance improvement, or very little.

For information and specific notes.ini parameters regarding transaction logging, refer to:

`http://www.lotus.com/ldd/today.nsf/Lookup/MoreLoggingVariables`

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this Redpaper.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 115. Note that some of the documents referenced here may be available in softcopy only.

- *Understanding LDAP - Design and Implementation*, SG24-4986
- *Tuning IBM @server xSeries Servers for Performance*, SG24-5287
- *IBM TotalStorage Disk Solutions for xSeries*, SG24-6874
- *Running the Linux 2.4 Kernel on IBM @server xSeries Servers*, REDP0121
- *Domino for IBM @server xSeries and BladeCenter Sizing and Performance Tuning*, REDP3851

Other publications

These publications are also relevant as further information sources:

- Stanfield, V., et al., *Linux System Administration*, Second Edition, Sybex Books, 2002, ISBN 0782141382
- Beck, M., et al., *Linux Kernel Internals*, Second Edition, Addison-Wesley Pub Co, 1997, ISBN 0201331438
- Musumeci, G-P., et al., *System Performance Tuning*, Second Edition, O'Reilly & Associates, 2002, ISBN 059600284X

Online resources

These Web sites and URLs are also relevant as further information sources:

- System Tuning Info for Linux Servers

  `http://people.redhat.com/alikins/system_tuning.html`
- Securing and Optimizing Linux (Red Hat 6.2)

  `http://www.faqs.org/docs/securing/index.html`
- Apache tuning information

  `http://perl.apache.org/docs/1.0/guide/performance.html`
- VMSTAT article

  `http://searchenterpriselinux.techtarget.com/tip/1,289483,sid39_gci952332,00.html`
- Linux 2.6 Kernels in Data Centers

  `http://www.osdl.org/docs/linux_2_6_datacenter_performance.pdf`

- Developer of ReiserFS

  http://www.namesys.com
- New features of V2.6 kernel

  http://www.infoworld.com/infoworld/article/04/01/30/05FElinux_1.html
- WebServing on 2.4 and 2.6

  http://www.ibm.com/developerworks/linux/library/l-web26/
- The man page about the **ab** command

  http://cmpp.linuxforum.net/cman-html/man1/ab.1.html
- Features of SUSE LINUX Enterprise Edition

  http://www.suse.com/us/business/products/server/sles/features.html
- RFC: Multicast

  http://www.ietf.org/rfc/rfc2365.txt
- RFC: Internet Control Message Protocol

  http://www.networksorcery.com/enp/RFC/Rfc792.txt
- RFC: Fault Isolation and Recovery

  http://www.networksorcery.com/enp/RFC/Rfc816.txt
- RFC: Type of Service in the Internet Protocol Suite

  http://www.networksorcery.com/enp/rfc/rfc1349.txt
- Performance and Tuning with OpenLDAP

  http://www.openldap.org/faq/data/cache/190.html
- RFC: TCP Extensions for Long-Delay Paths

  http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc1072.html
- RFC: TCP Extensions for High Performance

  http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc1323.html
- RFC: Extending TCP for Transactions -- Concepts

  http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc1379.html
- RFC: T/TCP -- TCP Extensions for Transactions

  http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc1644.html
- Internal functions of **ulimit** command

  http://www.scit.wlv.ac.uk/cgi-bin/mansec?2+getrlimit
- LOAD - Load and Performance Test Tools

  http://www.softwareqatest.com/qatweb1.html
- Apache HTTP Server Version 2.1 Documentation

  http://httpd.apache.org/docs-2.1/
- Information about Hyper-Threading

  http://www.intel.com/business/bss/products/hyperthreading/server/
- Information about EM64T

  http://www.intel.com/technology/64bitextensions/

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications, and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Tuning SUSE LINUX Enterprise Server on IBM @server xSeries Servers

Redpaper

Describes ways to tune the operating system

Introduces performance tuning tools

Covers key server applications

Linux is an open source operating system developed by people all over the world. The source code is freely available and can be used under the GNU General Public License. The operating system is made available to users in the form of distributions from companies such as SUSE LINUX. Whereas some desktop Linux distrubutions can be downloaded at no charge from the Web, the server versions typically must be purchased.

IBM has embraced Linux, and it is now recognized as an operating system suitable for enterprise-level applications running on IBM @server xSeries servers. Most enterprise applications are now available on Linux as well as Microsoft Windows, including file and print servers, database servers, Web servers, and collaboration and mail servers.

With the use in an enterprise-class server comes the need to monitor performance and, when necessary, tune the server to remove bottlenecks that affect users. This IBM Redpaper describes the methods you can use to tune SUSE LINUX Enterprise Server, tools you can use to monitor and analyze server performance, and key tuning parameters for specific server applications.

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