Tru64 UNIX

System Configuration and Tuning

Part Number: AA-RH9GC-TE

September 2002

Product Version: Tru64 UNIX Version 5.1B or higher

This manual describes how to tune Tru64 UNIX to improve operating system performance, and provides tuning recommendations for Oracle, Network File System and Web Server applications. It also includes tuning recommendations for Tru64 UNIX operating system components.

Hewlett-Packard Company
Palo Alto, California

Contents

2 Gathering System and Performance Information

5 Tuning Network File Systems

6 Tuning Internet Servers

7 Managing Application Performance

12 Managing Memory Performance

13 Managing CPU Performance

Glossary

Index

Figures

Tables

About This Manual

This manual contains information about tuning HP Tru64 UNIX for high performance and high availability. This manual also provides tuning recommendations for Oracle, Network File System (NFS), and Web server applications, and for specific Tru64 UNIX operating system components.

For Tru64 UNIX system administration, we recommend that you use the graphical user interface (GUI). This GUI is presented by SysMan, an application that is loaded by default when the Common Desktop Environment (CDE) software is loaded on your system. The SysMan applications are available in the Application Manager, which you can access from the CDE Front Panel.

Audience

This manual is intended for system administrators who are responsible for managing a Tru64 UNIX operating system. Administrators should have an in-depth knowledge of their applications and users, in addition to operating system concepts, commands, and utilities. Such an understanding is crucial to successfully tuning a system for better performance.

New and Changed Features

Additions and changes that have been made to the manual for this version of Tru64 UNIX include the following:

- This manual has been reorganized into three parts:
  - Introduction to System Tuning — Includes new hardware configuration information and diagrams.
  - Tuning by Application Type — Includes three ways to tune different applications: Oracle, Network File Systems, and Internet servers. This part includes detailed information about tunable attributes and system monitoring tools for each application type.
  - Tuning by Component — Includes tuning information on how to tune system resources, disk storage, network, file system, memory, and CPU. This part also includes detailed information about specific tunable attributes for each component.
- This manual describes only kernel subsystem attributes that can be used to improve system performance. All kernel subsystem attributes

are documented in the reference pages. See `sys_attrs`(5) for more information.

- Kernel subsystem attributes use only underscores instead of combinations of dashes and underscores.
- The Advanced File System (AdvFS) buffer cache is no longer used. AdvFS user and application data and metadata are now cached in the Unified Buffer Cache (UBC). See *Section 11.1.4* for more information.

  Documentation sections that described kernel subsystem attributes that related to the AdvFS buffer cache were removed.
- UNIX File System (UFS) supports smooth sync functionality and I/O throttling to improve UFS performance. See *Section 11.3.3.1* for more information.

Organization

This manual consists of thirteen chapters and a glossary:

Part I	**Introduction to System Tuning**
Chapter 1	Describes Tru64 UNIX hardware configuration and key terminology and concepts.
Chapter 2	Describes how to gather system information and diagnose performance problems.
Chapter 3	Describes how to access and modify kernel subsystems.
Part II	**Tuning by Application Type**
Chapter 4	Describes how to configure and tune an Oracle database application.
Chapter 5	Describes how to configure and tune a Network File System application.
Chapter 6	Describes how to configure and tune a Web Server application.
Chapter 7	Describes how to manage application performance.
Part III	**Tuning By Component**
Chapter 8	Describes how to manage system resource allocation.
Chapter 9	Describes how to monitor and tune disk storage performance.
Chapter 10	Describes how to monitor and tune network performance.
Chapter 11	Describes how to manage file system performance.
Chapter 12	Describes how to monitor and tune memory performance.
Chapter 13	Describes how to manage CPU performance.

Related Documents

The *System Administration* manual provides information on managing and monitoring your system. The *Programmer's Guide* provides information on the tools for programming on the Tru64 UNIX operating system. It also provides information on how to optimize the code used to create an application program, and how to optimize the results of the build process. The *Asynchronous Transfer Mode* manual contains information about tuning Asynchronous Transfer Mode (ATM).

The following Tru64 UNIX manuals also provide useful, relevant information:

- *Technical Overview*
- *Network Administration: Connections*
- *Logical Storage Manager*
- *AdvFS Administration*
- *Tru64 UNIX Version 5.1B QuickSpecs*
- *TruCluster Server Version 5.1B QuickSpecs*
- *Hardware Management*

Icons on Tru64 UNIX Printed Manuals

The printed version of the Tru64 UNIX documentation uses letter icons on the spines of the manuals to help specific audiences quickly find the manuals that meet their needs. (You can order the printed documentation from HP.) The following list describes this convention:

G	Manuals for general users
S	Manuals for system and network administrators
P	Manuals for programmers
R	Manuals for reference page users

Some manuals in the documentation help meet the needs of several audiences. For example, the information in some system manuals is also used by programmers. Keep this in mind when searching for information on specific topics.

The *Documentation Overview* provides information on all of the manuals in the Tru64 UNIX documentation set.

Reader's Comments

HP welcomes any comments and suggestions you have on this and other Tru64 UNIX manuals.

You can send your comments in the following ways:

- Fax: 603-884-0120 Attn: UBPG Publications, ZKO3-3/Y32
- Internet electronic mail: `readers_comment@zk3.dec.com`

  A Reader's Comment form is located on your system in the following location:

  `/usr/doc/readers_comment.txt`

Please include the following information along with your comments:

- The full title of the manual and the order number. (The order number appears on the title page of printed and PDF versions of a manual.)
- The section numbers and page numbers of the information on which you are commenting.
- The version of Tru64 UNIX that you are using.
- If known, the type of processor that is running the Tru64 UNIX software.

The Tru64 UNIX Publications group cannot respond to system problems or technical support inquiries. Please address technical questions to your local system vendor or to the appropriate HP technical support office. Information provided with the software media explains how to send problem reports to HP.

Conventions

The following conventions are used in this manual:

`#`	A number sign represents the superuser prompt.
`% cat`	Boldface type in interactive examples indicates typed user input.
file	Italic (slanted) type indicates variable values, placeholders, and function argument names.
[|] { | }	In syntax definitions, brackets indicate items that are optional and braces indicate items that are required. Vertical bars separating items inside brackets or braces indicate that you choose one item from among those listed.
⋮	A vertical ellipsis indicates that a portion of an example that would normally be present is not shown.
`cat`(1)	A cross-reference to a reference page includes the appropriate section number in parentheses. For example, `cat`(1) indicates that you can find information on the `cat` command in Section 1 of the reference pages.

Part 1

Introduction to System Tuning

1

Introduction to System Tuning

Tru64 UNIX offers various tools to monitor system performance. To improve your system performance, this manual offers tuning recommendations. Before you modify any system attributes, you should become familar with the following:

This chapter is an introduction to system tuning and describes the following:

- Hardware configuration (Section 1.1)
- Performance terminology and concepts (Section 1.2)
- Disk storage resources (Section 1.3)
- Network resources (Section 1.4)
- File system resources (Section 1.5)
- Memory resources (Section 1.6)
- CPU resources (Section 1.7)
- Identifying a resource model for your workload (Section 1.8)
- Most commonly tuned kernel subsystems (Section 1.9)

1.1 Hardware Configuration

A **configuration** consists of system, disk storage, and network hardware, in addition to the operating system and application software. Different configurations provide various amounts of CPU power, memory resources, I/O performance, and storage capacity. Use the configuration guidelines in this manual to choose the configuration that is appropriate for your workload, performance, and availability needs.

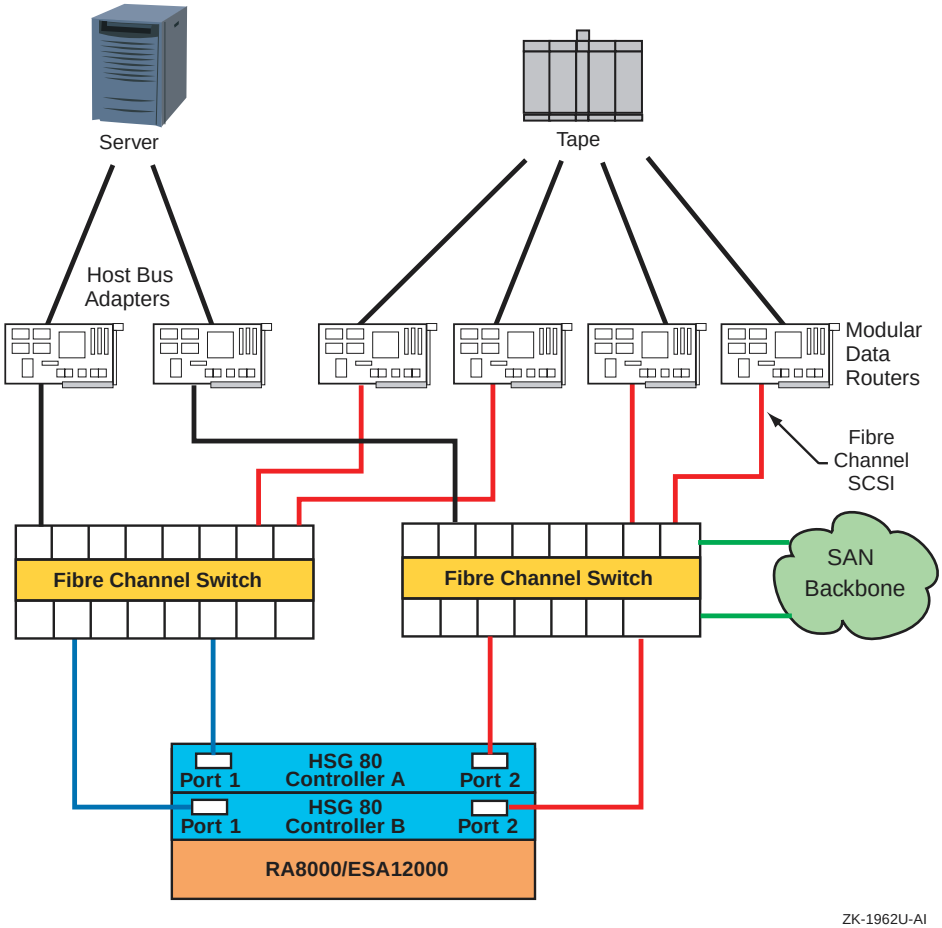
After you **configure** the system, you may be able to **tune** the operating system to improve performance. Tuning usually involves modifying the kernel by changing the default values of **attributes**, which affect the behavior and performance of kernel subsystems.

The following sections provide some background information about how the CPU, memory, and I/O configuration affect performance. See the *Tru64 UNIX Version 5.1B QuickSpecs* and the *Technical Overview* for information about hardware and operating system performance features.

1.1.1 Hardware Configuration Overview

We recommend that you create a diagram for your own hardware configuration to help you understand the hardware environment of your system. Figure 1–1 shows a sample hardware configuration. It includes the major hardware components of a system that affect performance, such as the number of CPUs, host bus adapters, network interface cards, Fibre Channel switches and connections, and storage arrays.

Figure 1–1: Mapping Out Your Hardware Configuration



1.2 Performance Terminology and Concepts

System performance depends on an efficient utilization of system **resources**, which are the hardware and software components available to users or

applications. A system must perform well under the normal workload exerted on the system by applications and users.

Because workloads change over time (for example, running additional applications), a system must be **scalable**, which refers to a system's ability to utilize additional hardware resources with a predictable impact on performance. **Scalability** can also refer to the ability of a system to absorb an increase in workload without a significant performance degradation.

A performance problem in a specific area of the configuration is called a **bottleneck**. A bottleneck can occur if the workload demands more from a resource than its **capacity**, which is the maximum theoretical throughput of a system resource.

Performance is often described in terms of two rates. **Bandwidth** is the rate at which an I/O subsystem or component can transfer bytes of data. Bandwidth is often called the **transfer rate**. Bandwidth is especially important for applications that perform large sequential data transfers.

Throughput is the rate at which an I/O subsystem or component can perform I/O operations. Throughput is especially important for applications that perform many small I/O operations.

Performance is also measured in terms of **latency**, which is the amount of time to complete a specific operation. Latency is often called **delay**. High-performance systems require a low latency time. I/O latency is measured in milliseconds; memory latency is measured in nanoseconds. Memory latency depends on the memory bank configuration and the amount of memory.

Disk performance is often described in terms of **disk access time**, which is a combination of the **seek time**, the amount of time for a disk head to move to a specific disk track, and the **rotational latency**, which is the amount of time for a disk to rotate to a specific disk sector.

A data transfer can consist of file-system data or **raw I/O**, which is I/O to a disk or disk partition that does not contain a file system. Raw I/O bypasses buffers and caches, and it may provide better performance, in some cases, than file system I/O. Raw I/O is often used by the operating system and by database application software.

Data transfers also have different access patterns. A **sequential access pattern** is an access pattern in which data is read from or written to contiguous (adjacent) blocks on a disk. A **random access pattern** is an access pattern in which data is read from or written to blocks in different (usually nonadjacent) locations on a disk.

1.3 Disk Storage Resources

Disk storage configurations vary greatly, so you must determine which configuration will meet the performance and availability needs of your applications and users.

Disk storage configurations can consist of single disks with traditional discrete **disk partitions**. However, you may want to use the Logical Storage Manager (LSM) to manage large amounts of disk storage. LSM enables you to set up a shared pool of storage, and also provides high-performance and high-availability features, such as RAID support.

Storage configurations can also include **hardware RAID** subsystems, which greatly expand the number of disks that can be connected to a single I/O bus and provide many high-performance and high-availability features, including RAID support and write-back caches. There are various types of hardware RAID subsystems that are suitable for different environments.

Host bus adapters, RAID controllers, and disks have various performance features and support Fibre Channel and different parallel Small Computer System Interface (**SCSI**) variants. Fibre Channel and SCSI are device and interconnect technologies that continue to evolve in terms of high performance, availability, and configuration flexibility. The following sections discuss disk storage resources in more detail:

- See Section 1.3.1 for more information about RAID functionality.
- See Section 1.3.2 for more information about SCSI.
- See Section 1.3.3 for more information about Fibre Channel.
- See Chapter 9 for more information about storage configurations.

1.3.1 RAID Technology

You can use redundant array of independent disks (**RAID**) technology in a storage configuration for high performance and high data availability. You can obtain RAID functionality by using Logical Storage Manager (LSM) or a hardware-based RAID subsystem.

There are four primary RAID levels:

- **RAID0** — Also known as data or disk **striping**, RAID0 divides data into blocks and distributes the blocks across multiple disks in an array, which improves throughput. Striping does not provide disk data availability.
- **RAID1** — Also known as data or disk **mirroring**, RAID1 maintains identical copies of data on different disks in an array. Duplicating data on different disks provides high data availability and improves disk read performance. You can combine RAID1 with RAID0 to mirror striped data or disks.

- **RAID3** — A type of **parity RAID**, RAID3 divides data blocks and distributes the data across a disk array, providing parallel access to data and increasing bandwidth. RAID3 also provides data availability by placing redundant parity information on a separate disk, which is used to regenerate data if a disk in the array fails.
- **RAID5** — A type of parity RAID, RAID5 distributes data blocks across disks in an array. RAID5 allows independent access to data and can handle simultaneous I/O operations, which improves throughput. RAID5 provides data availability by distributing redundant parity information across the array of disks. The parity information is used to regenerate data if a disk in the array fails.

In addition, high-performance RAID controllers support **dynamic parity RAID** (also called **adaptive RAID3/5**), which combines the benefits of RAID3 and RAID5 to improve disk I/O performance for a wide variety of applications. Dynamic parity RAID dynamically adjusts, according to workload needs, between data transfer-intensive algorithms and I/O operation-intensive algorithms.

It is important to understand that RAID performance depends on the state of the devices in the RAID subsystem. There are three possible states:

- Steady state (no failures)
- Failure state (one or more disks have failed)
- Recovery state (subsystem is recovering from failure)

Table 1–1 compares the performance features and degrees of availability for the different RAID levels.

Table 1–1: RAID Level Performance and Availability Comparison

RAID Level	Performance Feature	Degree of Availability
RAID0 (striping)	Balances I/O load and improves throughput	Lower than single disk
RAID1 (mirroring)	Improves read performance, but degrades write performance	Highest
RAID0+1	Balances I/O load and improves throughput, but degrades write performance	Highest
RAID3	Improves bandwidth, but performance may degrade if multiple disks fail	Higher than single disk

Table 1–1: RAID Level Performance and Availability Comparison (cont.)

RAID Level	Performance Feature	Degree of Availability
RAID5	Improves throughput, but performance may degrade if multiple disks fail	Higher than single disk
Dynamic parity RAID (RAID3/5)	Improves bandwidth and throughput, but performance may degrade if multiple disks fail	Higher than single disk

There are many variables to consider when choosing a RAID configuration:

- Not all RAID products support all RAID levels.

  For example, only high-performance RAID controllers support dynamic parity RAID.

- RAID products provide different performance features.

  For example, only RAID controllers support write-back caches and relieve the CPU of the I/O overhead.

- Some RAID configurations are more cost-effective than others.

  In general, LSM provides more cost-effective RAID functionality than hardware RAID subsystems. In addition, parity RAID provides data availability at a cost that is lower than RAID1 (mirroring), because mirroring n disks requires $2n$ disks.

- Data recovery rates depend on the RAID configuration.

  For example, if a disk fails, it is faster to regenerate data by using a mirrored copy than by using parity information. In addition, if you are using parity RAID, I/O performance declines as additional disks fail.

See Chapter 9 for more information about RAID configurations.

1.3.2 SCSI Concepts

The most common type of SCSI is **parallel SCSI**, which supports SCSI variants that provide you with a variety of performance and configuration options. The SCSI variants are based on data path (narrow or wide), bus speed (Slow, Fast, Ultra, Ultra2, or Ultra3), and transmission method (single-ended or differential). These variants determine the bus bandwidth and the maximum allowable SCSI bus length.

Serial SCSI is the next generation of SCSI. Serial SCSI reduces parallel SCSI's limitations on speed, distance, and connectivity (number of devices on the bus), and also provides availability features like hot swap and fault tolerance.

Fibre Channel is an example of serial SCSI. A high-performance I/O bus that supports multiple protocols (SCSI, IPI, FIPS60, TCP/IP, HIPPI, and so forth), Fibre Channel is based on a network of intelligent switches. Link speeds are available up to 100 MB/sec in full-duplex mode.

The following sections describe parallel SCSI concepts in detail.

1.3.2.1 Data Paths

Disks, host bus adapters, I/O controllers, and storage enclosures support a specific **data path**. The data path and the bus speed determine the actual bandwidth for a bus. There are two data paths available:

- **Narrow data path**

  Specifies an 8-bit data path. The performance of this mode is limited. SCSI bus specifications restrict the number of devices on a narrow SCSI bus to eight.

- **Wide data path**

  Specifies a 16-bit data path for Slow, Fast SCSI, UltraSCSI, Ultra2, and Ultra3. This mode increases the amount of data that is transferred in parallel on the bus. SCSI bus specifications restrict the number of devices on a wide bus to 16.

  Disks and host bus adapters that use a wide data path can provide nearly twice the bandwidth of disks and adapters that use a narrow data path. Wide devices can greatly improve I/O performance for large data transfers.

Most current disks support wide data paths. Older disks have versions that support wide and narrow data paths. Devices with different data paths (or transmission methods) cannot be directly connected on a single bus. You must use a SCSI signal converter (for example, a DWZZA or DWZZB) or an **UltraSCSI** extender (for example, a DWZZC or DWZZH [SCSI hub]) to connect devices with different data paths.

1.3.2.2 SCSI Bus Speeds

The **SCSI bus speed**, also called the transfer rate or bandwidth, is the number of transfers per second. Faster bus speeds provide the best performance. Both bus speed and the data path (narrow or wide) determine the actual bus bandwidth (number of bytes transferred per second).

Not all devices support all bus speeds. To set the bus speed on a host bus adapter, use either console commands or the Loadable Firmware Update (LFU) utility, depending on the type of adapter. See the *TruCluster Server Software Version 5.1B QuickSpecs* for information about SCSI device support.

Table 1–2 shows the available bus speeds.

Table 1–2: SCSI Bus Speeds

Bus Speed	Maximum Transfer Rate (million transfers/sec)	Maximum Byte Transfer Rate: Narrow (MB/sec)	Maximum Byte Transfer Rate: Wide (MB/sec)
Ultra3	80	80	160
Ultra2	40	40	80
UltraSCSI	20	20	40
Fast SCSI	10	10	20
Slow	5	5	10

HP's implementation of Ultra3 is compatible with and compliant to the Ultra 160/m implementation of the Ultra3 SCSI specification.

Fast SCSI, also called **Fast10**, is an extension to the SCSI-2 specification. It uses the fast synchronous transfer option, enabling I/O devices to attain high peak-rate transfers in synchronous mode.

UltraSCSI, also called **Fast20**, is a high-performance, extended version of SCSI-2 that reduces many performance and configuration deficiencies of Fast SCSI. Compared to Fast SCSI bus speed, UltraSCSI doubles the bandwidth and configuration distances, but with no increase in cost. UltraSCSI also provides faster transaction times and faster, more accurate data analysis. All UltraSCSI components are backward compatible with regular SCSI-2 components.

1.3.2.3 Transmission Methods

The **transmission method** for a bus refers to the electrical implementation of the SCSI specification. Supported transmissions methods include:

- **Single-ended (SE) SCSI**

  Used to connect devices that are usually located within the same cabinet. Single-ended SCSI usually requires short cable lengths.

  A single-ended SCSI bus uses one data lead and one ground lead for the data transmission. A single-ended receiver looks at only the signal wire as the input. The transmitted signal arrives at the receiving end of the bus on the signal wire slightly distorted by signal reflections. The length and loading of the bus determine the magnitude of this distortion. Therefore, the single-ended transmission method is economical, but it is more susceptible to noise than the differential transmission method and requires short cables.

- **Differential SCSI**

  Used to connect devices that are up to 25 meters apart.

A differential SCSI bus uses two wires to transmit a signal. The two wires are driven by a differential driver that places a signal on one wire (+SIGNAL) and another signal that is 180 degrees out of phase (-SIGNAL) on the other wire. The differential receiver generates a signal output only when the two inputs are different. Because signal reflections are virtually the same on both wires, they are not seen by the receiver, which notices only differences on the two wires. The differential transmission method is less susceptible to noise than single-ended SCSI and enables you to use long cables.

You cannot directly connect SE and differential devices on the same bus. To connect SE and differential devices on the same bus you must use an UltraSCSI extender.

- **Low Voltage Differential (LVD) SCSI**

  Same as differential SCSI, but uses low voltage and you can directly connect SE and LVD SCSI drives on the same SCSI bus.

  When SE and LVD SCSI devices are connected on the same SCSI bus, performance is limited to SE SCSI operation (40 MB/sec) for all devices on the SCSI bus (for that particular SCSI channel). According to the rules of SCSI, to maintain a true LVD SCSI bus and its associated performance, only LVD SCSI drives can be on the same LVD SCSI channel. However, this does not prevent the support of dedicated SE channels and dedicated LVD channels, all on a single array controller.

  Ultra2 and Ultra3 devices operate on the LVD electrical platform. When Ultra2 and Ultra3 devices are connected on the same Ultra3 SCSI bus, the Ultra2 devices will transfer data up to 80 MB/sec, while the Ultra3 devices will transfer data up to 160 MB/sec. If the SCSI bus is only capable of supporting Ultra2, all LVD devices will have a maximum transfer of 80 MB/sec.

1.3.2.4 Extending UltraSCSI Bus Segments

UltraSCSI devices can be either single-ended or differential. Because of UltraSCSI's high bus speed, single-ended UltraSCSI signals cannot maintain their strength and integrity over the same distance as single-ended Fast SCSI signals. Therefore, UltraSCSI technology uses **bus segments** and **bus extenders** so that systems and storage can be configured over long distances.

An UltraSCSI bus extender joins two bus segments together without any impact on SCSI protocol. A bus segment is defined as an unbroken electrical path consisting of conductors (in cables or backplanes) and connectors. Every UltraSCSI bus segment must have two terminators, one at each end of the bus segment. Therefore, an UltraSCSI bus segment corresponds to an entire bus in Fast SCSI. The SCSI domain is the collection of SCSI devices on all

the bus segments. As with a Fast SCSI bus, an UltraSCSI bus segment can only support devices of the same type (single-ended or differential).

Although UltraSCSI components allow an UltraSCSI domain to extend for longer distances than a Fast SCSI bus, there are still limits. Also, because the use of bus expanders allows UltraSCSI domains to look like a tree instead of a straight line, the concept of bus length must be replaced with the concept of the UltraSCSI domain diameter.

1.3.2.5 SCSI Bus Length and Termination

There is a limit to the length of the cables in a SCSI bus. The maximum cable length depends on the bus speed and the transmission method (single-ended or differential). The total cable length for a physical bus or UltraSCSI bus segment is calculated from one terminated end to the other.

In addition, each SCSI bus or bus segment must be terminated only at each end. Improper bus termination and lengths are a common cause of bus malfunction.

If you are using devices that have the same transmission method and data path (for example, wide and differential), a bus will consist of only one physical bus (or multiple bus fragments in the case of UltraSCSI). If you have devices with different transmission methods, you will have both single-ended and differential physical buses or bus segments, each of which must be terminated only at both ends and adhere to the rules on bus length.

Table 1–3 shows the maximum bus lengths for different bus speeds and transmission methods.

Table 1–3: SCSI Bus and Segment Lengths

Bus Speed	Transmission Method	Maximum Bus or Segment Length
Slow	Single-ended	6 meters
Fast	Single-ended	3 meters
Fast	Differential	25 meters
Ultra	Differential	25 meters
Ultra	Single-ended	1.5 meters (daisy-chain configuration in which devices are spaced less than 1 meter apart)

Table 1–3: SCSI Bus and Segment Lengths (cont.)

Bus Speed	Transmission Method	Maximum Bus or Segment Length
Ultra	Single-ended	4 meters (daisy-chain configuration in which devices are spaced more than 1 meter apart)
Ultra	Single-ended	20 meters (point to point configuration in which devices are only at the ends of the bus segment)

Note that the total length of a physical bus must include the amount of cable that is located inside each system and disk storage shelf. This length varies, depending on the device. For example, the length of cable inside a BA350, BA353, or BA356 storage shelf is approximately 1.0 meter.

1.3.3 Fibre Channel

Fibre Channel supports multiple protocols over the same physical interface. Fibre Channel is primarily a protocol-independent transport medium; therefore, it is independent of the function for which you use it.

Tru64 UNIX uses the Fibre Channel Protocol (FCP) for SCSI to use Fibre Channel as the physical interface.

Fibre Channel, with its serial transmission method, overcomes the limitations of parallel SCSI by providing:

- Data rates of 100 MB/sec, 200 MB/sec, and 400 MB/sec
- Support for multiple protocols
- Better scalability
- Improved reliability and availability

Fibre Channel uses an extremely high-transmit clock frequency to achieve the high data rate. Using optical fiber transmission lines allows the high-frequency information to be sent up to 40 kilometers (24.85 miles), which is the maximum distance between transmitter and receiver. Copper transmission lines may be used for shorter distances.

The following sections describe Fibre Channel in more detail:

- Fibre Channel topologies (Section 1.3.3.1)
- Fibre Channel topology comparison (Section 1.3.3.2)
- Zoning (Section 1.3.3.3)

1.3.3.1 Fibre Channel Topologies

Fibre Channel supports three different interconnect topologies:

- Point-to-point (Section 1.3.3.1.1)
- Fabric (Section 1.3.3.1.2)
- Arbitrated loop (Section 1.3.3.1.3)

Note

Although you can interconnect an arbitrated loop with the fabric, hybrid configurations are not currently supported; therefore, those configurations are not discussed in this manual.

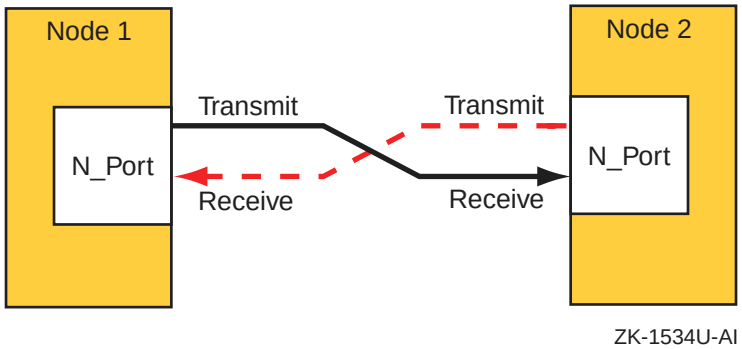
1.3.3.1.1 Point-to-Point Topology

The point-to-point topology is the simplest Fibre Channel topology. In a point-to-point topology, one **N_Port** is connected to another N_Port by a single **link**.

Because all frames transmitted by one N_Port are received by the other N_Port, and in the same order in which they were sent, frames require no routing.

Figure 1–2 shows an example point-to-point topology.

Figure 1–2: Point-to-Point Topology



1.3.3.1.2 Fabric Topology

The **fabric** topology provides more connectivity than point-to-point topology. The fabric topology can connect up to 2^{24} ports.

The fabric examines the destination address in the frame header and routes the frame to the destination node.

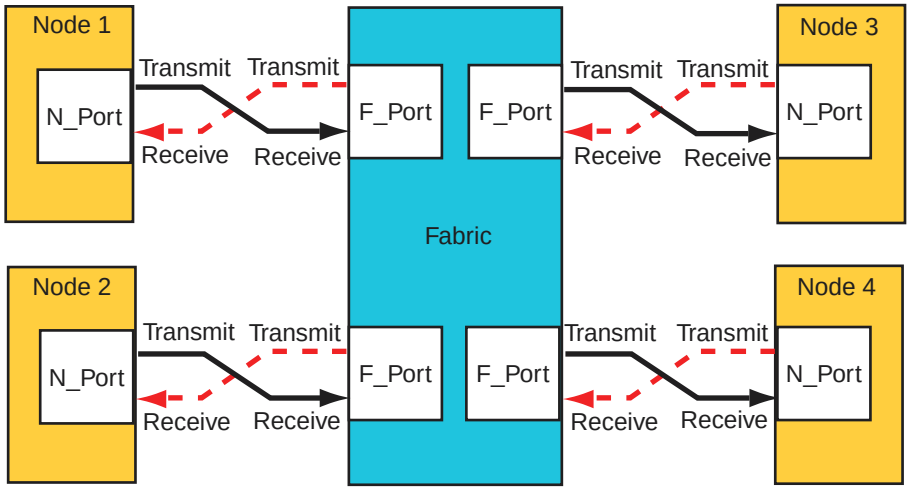
A fabric may consist of a single switch, or there may be several interconnected switches (up to three interconnected switches are supported). Each switch contains two or more fabric ports (**F_Ports**) that are internally connected by the fabric switching function, which routes the frame from one F_Port to another F_Port within the switch. Communication between two switches is routed between two expansion ports (**E_Ports**).

When an N_Port is connected to an F_Port, the fabric is responsible for the assignment of the Fibre Channel address to the N_Port attached to the fabric. The fabric is also responsible for selecting the route a frame will take, within the fabric, to be delivered to the destination.

When the fabric consists of multiple switches, the fabric can determine an alternate route to ensure that a frame gets delivered to its destination.

Figure 1–3 shows an example fabric topology.

Figure 1–3: Fabric Topology



1.3.3.1.3 Arbitrated Loop Topology

In an **arbitrated loop** topology, **frames** are routed around a loop set up by the links between the nodes. The hub maintains loop continuity by bypassing a **node** when the node or its cabling fails, when the node is powered down, or when the node is removed for maintenance. The hub is transparent to the protocol. It does not consume any Fibre Channel arbitrated loop addresses so it is not addressable by a Fibre Channel arbitrated loop port.

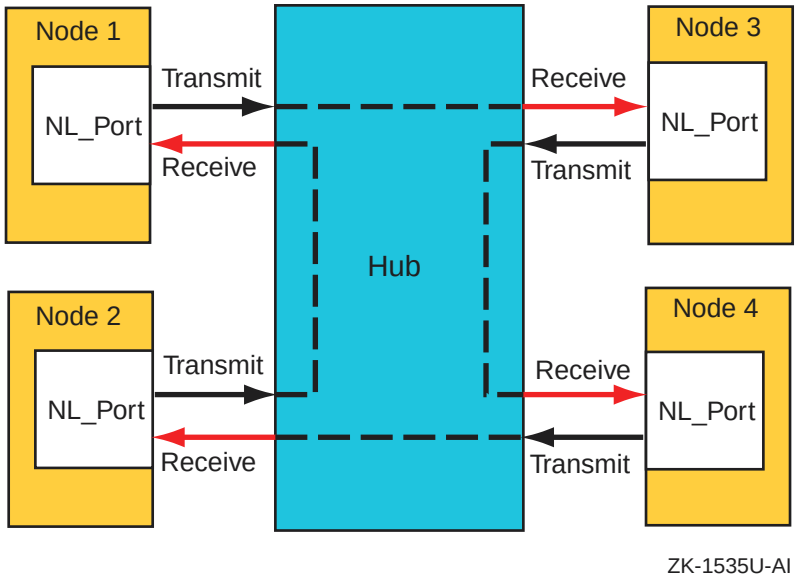
The nodes arbitrate to gain control (become master) of the loop. After a node becomes the master, the nodes select (by way of setting bits in a bitmask)

their own Arbitrated Loop Physical Address (AL_PA). The **AL_PA** is used to address nodes on the loop. The AL_PA is dynamic and can change each time the loop is initialized, a node is added or removed, or at any other time that an event causes the membership of the loop to change. When a node is ready to transmit data, it transmits Fibre Channel primitive signals that include its own identifying AL_PA.

In the arbitrated loop topology, a node port is called an **NL_Port** (node loop port), and a fabric port is called an **FL_Port** (fabric loop port).

Figure 1–4 shows an example of an arbitrated loop topology.

Figure 1–4: Arbitrated Loop Topology



1.3.3.2 Fibre Channel Topology Comparison

This section compares and contrasts the fabric and arbitrated loop topologies, and describes why you might choose to use them.

When compared with the fabric (switched) topology, arbitrated loop is a lower cost, and lower performance, alternative. Arbitrated loop reduces Fibre Channel cost by substituting a lower-cost, often nonintelligent and unmanaged hub, for a more expensive switch. The hub operates by collapsing the physical loop into a logical star. The cables, associated connectors, and allowable cable lengths are similar to those of a fabric. Arbitrated loop supports a theoretical limit of 127 nodes in a loop. Arbitrated loop nodes are self-configuring and do not require Fibre Channel address switches.

Arbitrated loop provides reduced cost at the expense of bandwidth; all nodes in a loop share the bandwidth (100 MB/sec per loop), and bandwidth degrades slightly as nodes and cables are added. Nodes on the loop see all traffic on the loop, including traffic between other nodes. The hub can include port-bypass functions that manage movement of nodes on and off the loop. For example, if the port bypass logic detects a problem, the hub can remove that node from the loop without intervention. Data availability is then preserved by preventing the down time associated with node failures, cable disconnections, and network reconfigurations. However, traffic caused by node insertion and removal, errors, and so forth, can cause temporary disruption on the loop.

Although the fabric topology is more expensive, it provides both increased connectivity and higher performance; switches provide a full-duplex 100 (200) MB/sec point-to-point connection to the fabric. Switches also provide improved performance and scaling because nodes on the fabric see only data destined for themselves, and individual nodes are isolated from reconfiguration and error recovery of other nodes within the fabric. Switches can provide management information about the overall structure of the Fibre Channel fabric, which may not be the case for an arbitrated loop hub.

Table 1–4 compares the fabric and arbitrated loop topologies.

Table 1–4: Fibre Channel Fabric and Arbitrated Loop Comparison

When to Use Arbitrated Loop	When to Use Fabric
In clusters of up to two members	In clusters of more than two members
In applications where low total solution cost and simplicity are key requirements	In multinode cluster configurations when possible temporary traffic disruption due to reconfiguration or repair is a concern
In applications where the shared bandwidth of an arbitrated loop configuration is not a limiting factor	In high bandwidth applications where a shared arbitrated loop topology is not adequate
In configurations where expansion and scaling are not anticipated	In cluster configurations where expansion is anticipated and requires performance scaling

1.3.3.3 Zoning

This section provides a brief overview of zoning.

A **zone** is a logical subset of the Fibre Channel devices that are connected to the fabric. Zoning allows partitioning of resources for management and access control. In some configurations, it may provide for more efficient use of hardware resources by allowing one switch to serve multiple clusters or even multiple operating systems. **Zoning** entails splitting the fabric into zones, where each zone is essentially a virtual fabric.

Zoning may be used:

- When you want to set up barriers between systems of different operating environments or uses; for example, to allow two clusters to utilize the same switch.
- To create test areas that are separate from the rest of the fabric.
- To provide better utilization of a switch by reducing the number of unused ports.

Note

Any initial zoning must be made before connecting the host bus adapters and the storage to the switches. However, after zoning is configured, changes can be made dynamically.

1.3.3.3.1 Switch Zoning Versus Selective Storage Presentation

Switch zoning and the **selective storage presentation (SSP)** feature of the HSG80 controllers have similar functions.

Switch zoning controls which servers can communicate with each other and each storage controller host port. SSP controls which servers will have access to each storage unit.

Switch zoning controls access at the storage system level; SSP controls access at the storage unit level.

The following configurations require zoning or selective storage presentation:

- When you have a TruCluster Server cluster in a storage array network (SAN) with other standalone systems (UNIX or non-UNIX), or other clusters.
- Any time you have Windows NT or Windows 2000 in the same SAN with Tru64 UNIX. (Windows NT or Windows 2000 must be in a separate switch zone.)
- The SAN configuration has more than 64 connections to an RA8000, ESA12000, MA6000, MA8000, or EMA12000.

The use of selective storage presentation is the preferred way to control access to storage (so zoning is not required).

1.3.3.3.2 Types of Zoning

There are two types of zoning, soft and hard:

- **Soft zoning** is a software implementation that is based on the **Simple Name Server (SNS)** enforcing a zone. Zones are defined by either the node or port **World Wide Names (WWN)**, or the domain and port numbers in the form of D,P, where D is the domain and P is the physical port number on the switch.

  A host system requests a list of all adapters and storage controllers that are connected to the fabric. The name service provides a list of all ports that are in the same zone or zones as the requesting host bus adapter.

  Soft zoning only works if all hosts honor it; it does not work if a host is not programmed to allow for soft zoning. For example, if a host tries to access a controller that is outside the zone, the switch does not prevent the access.

  Tru64 UNIX honors soft zoning and does not attempt to access devices outside the zone.

  If you have used the WWN to define the zone and replace a KGPSA host bus adapter, you must modify the zone configuration and SSP because the node WWN has changed.

- With hard zoning, zones are enforced at the physical level across all fabric switches by hardware blocking of the Fibre Channel frames. Hardware zone definitions are in the form of D,P, where D is the domain and P is the physical port number on the switch. An example might be 1,2 for switch 1, port 2.

  If a host attempts to access a port that is outside its zone, the switch hardware blocks the access.

  You must modify the zone configuration when you move any cables from one port to another within the zone.

If you want to guarantee that there is no access outside any zone, either use hard zoning, or use operating systems that state they support soft zoning.

Table 1–5 lists the types of zoning that are supported on each of the supported Fibre Channel switches.

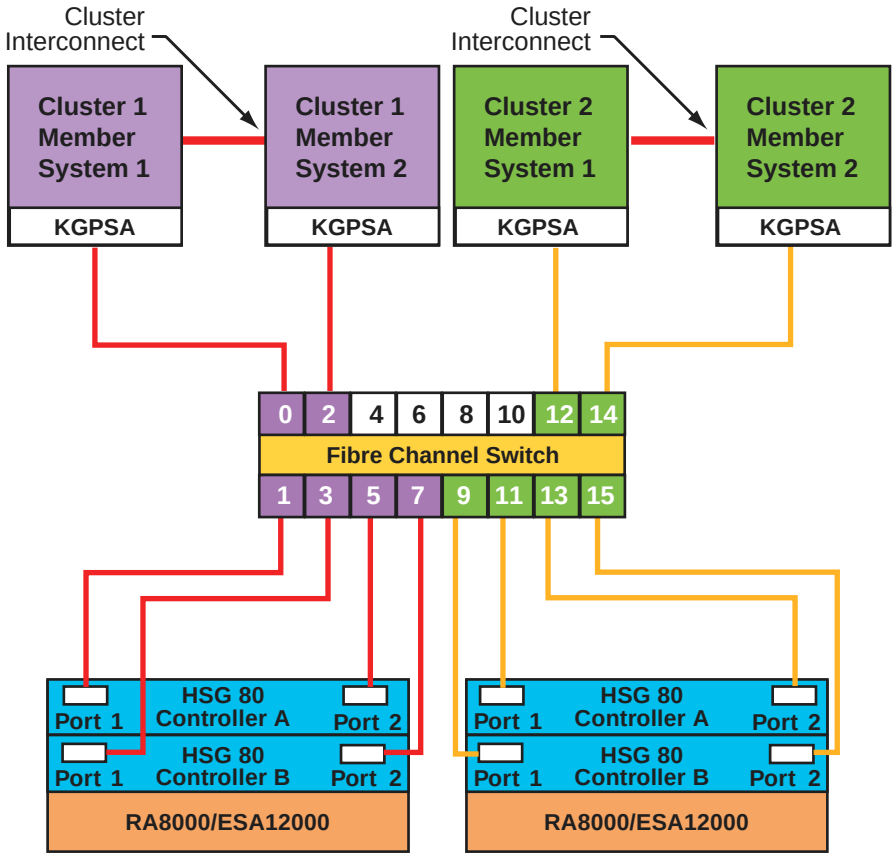
Table 1–5: Type of Zoning Supported by Switches

Switch Type	Type of Zoning Supported
DS-DSGGA	Soft
DS-DSGGB	Soft and Hard
DS-DSGGC	Soft and Hard

1.3.3.3.3 Zoning Example

Figure 1–5 shows a sample configuration using zoning. This configuration consists of two independent zones with each zone containing an independent cluster.

Figure 1–5: A Simple Zoned Configuration



For information on setting up zoning, see the SAN Switch Zoning documentation that is provided with the switch.

See the *Cluster Hardware Configuration* manual for more information.

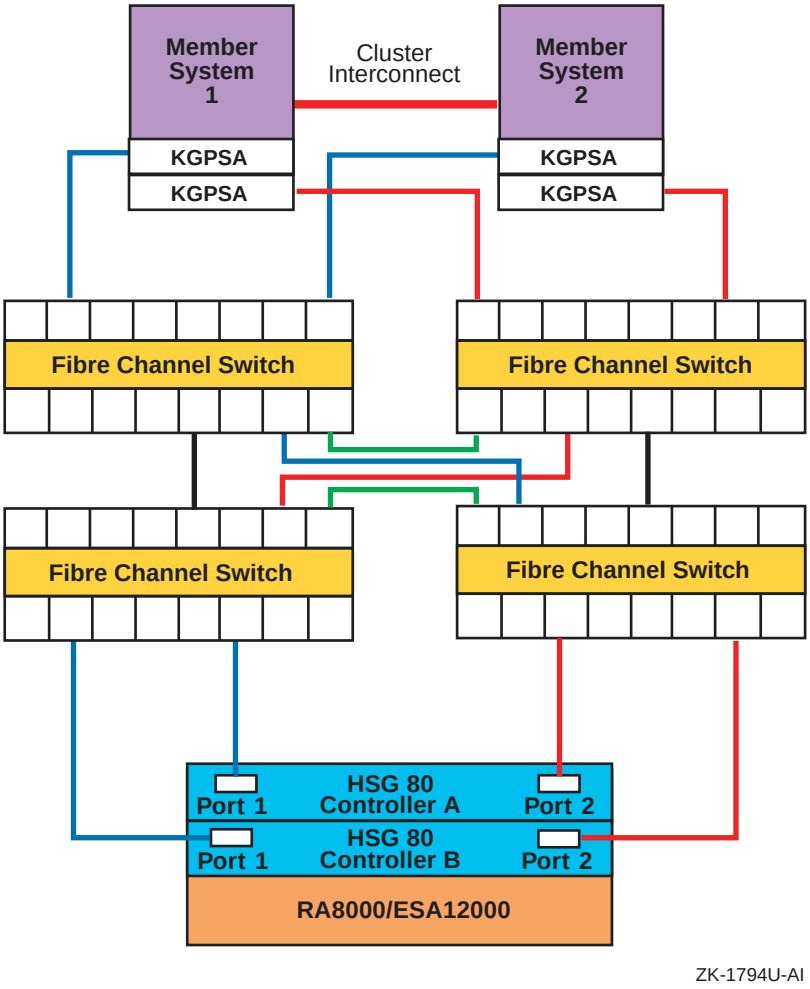
1.3.3.4 Cascaded Switches

Multiple switches may be connected to each other to form a network of switches, or **cascaded switches**.

A cascaded switch configuration, which allows for network failures up to and including the switch without losing a data path to a SAN connected node, is called a **mesh** or **meshed** fabric.

Figure 1–6 shows an example meshed resilient fabric with four cascaded interconnected switches. This configuration will tolerate multiple data path failures, and is an NSPOF (no single point of failure) configuration.

Figure 1–6: Meshed Resilient Fabric with Four Cascaded Switches



Note

If you lose an ISL, the communication can be routed through another switch to the same port on the other controller. This can constitute the maximum allowable two hops.

See the *Cluster Hardware Configuration* manual for more information on Fibre Channel.

1.4 Network Resources

Systems support various networks and network adapters that provide different performance features. For example, an Asynchronous Transfer Mode (ATM) high-performance network is ideal for applications that need the high speed and the low latency (switched, full-duplex network infrastructure) that ATM networks provide.

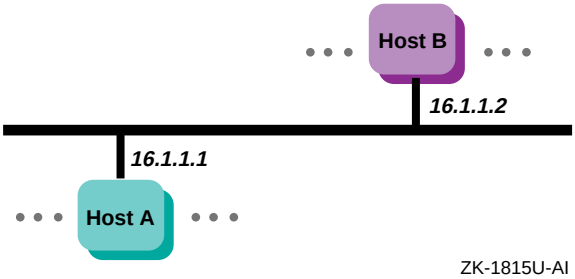
In addition, you can configure multiple network adapters or use NetRAIN to increase network access and provide high network avialability.

Your system is connected to the network through a **Network Interface Card (NIC)** (which is also called a **network interface** or **network adapter**). End systems or hosts can have the following interface options:

- Single interface in a subnet
- Multiple interfaces in a subnet
- Multiple interfaces with automatic failover (NetRAIN)
- Multiple aggregated interfaces (link aggregation)

Routers typically have multiple interfaces, each connected to a different subnet. Figure 1–7 shows a network with two hosts, Host A and Host B, each with a single network interface in a subnet.

Figure 1–7: Single Interface Configuration



The following sections discuss network resources that are important to improve system performance.

1.4.1 Network Subsystem

Most resources used by the network subsystem are allocated and adjusted dynamically; however, there are some tuning guidelines that you can use to improve performance, particularly with systems that are Internet servers, including Web, proxy, firewall, and gateway servers.

Network performance is affected when the supply of resources is unable to keep up with the demand for resources. The following two conditions can cause this to occur:

- A problem with one or more hardware or software network components
- A workload (network traffic) that consistently exceeds the capacity of the available resources, although everything appears to be operating correctly

Neither of these problems are network tuning issues. In the case of a problem on the network, you must isolate and eliminate the problem. In the case of high network traffic (for example, the hit rate on a Web server has reached its maximum value while the system is 100 percent busy), you must either redesign the network and redistribute the load, reduce the number of network clients, or increase the number of systems handling the network load.

1.4.2 Using Redundant Networks

Network connections may fail because of a failed network interface or a problem in the network itself. You can make the network connection highly available by using redundant network connections. If one connection becomes unavailable, you can still use the other connection for network access. Whether you can use multiple networks depends on the application, network configuration, and network protocol.

You can also use **NetRAIN (redundant array of independent network adapters)** to configure multiple interfaces on the same LAN segment into a single interface, and to provide failover support for network adapter and network connections. One interface is always active while the other interfaces remain idle. If the active interface fails, an idle interface is brought on line within less than 10 seconds.

NetRAIN supports only Ethernet and FDDI, see Section 1.4.3 for more information about NetRAIN.

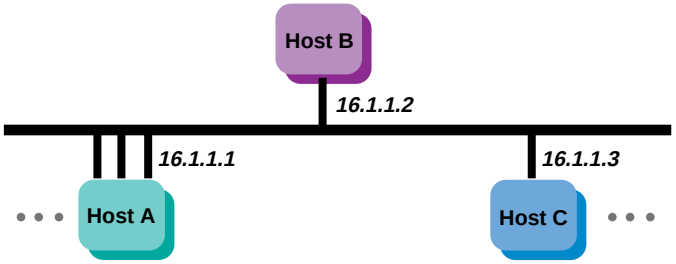
See `nr`(7) for more information about NetRAIN. See the *Network Administration: Connections* guide for information about network configuration. See Chapter 10 for information about improving network performance.

1.4.3 NetRAIN

The Redundant Array of Independent Network Adaptors (NetRAIN) interface provides a mechanism to protect against certain kinds of network connectivity failures.

NetRAIN integrates multiple network interfaces on the same local area network (LAN) segment into a single virtual interface called a NetRAIN set. One network interface in the set is always active while the others remain idle. If the active interface fails, one of the idle set members comes on line with the same IP address within an adjustable failover time period. Figure 1–8 shows Host A with three interfaces that are part of a NetRAIN set. The NetRAIN virtual interface is assigned the address 16.1.1.1.

Figure 1–8: Mulitple Interfaces



NetRAIN monitors the status of its network interfaces with the Network Interface Failure Finder (NIFF), a tool used to detect and report possible network failures. This tool can be used independently of NetRAIN. For more information about NIFF, see `NIFF`(7).

1.4.4 Routing

All systems (hosts and routers) connected to a network must be configured to support network routing in order to communicate with other systems on other networks. A **route** is the path a packet takes through a network from one system to another. As such it enables you to communicate with other systems on other networks. Routes are stored on each system in the routing tables or routing database. Each route entry consists of the following:

- A destination address (either a network or a host)
- The address of the next hop from your system to the destination
- The address of your system on the network if the route is through an interface
- A network interface (for example, `tu0` and `fta0`)

- Metrics (for example, hop count and MTU)

Additional routes might be added to your routing tables based on Internet Control Message Protocol (ICMP) redirect messages. These are messages from routers to hosts that tell the host to forward traffic to another router on the local network.

1.4.5 LAG Interface

Link aggregation (LAG) interfaces provide higher availability, fault tolerance, and load sharing on systems that contain multiple network adapters. Link aggregation, or trunking, also enables administrators to combine one or more physical Ethernet NICs and create a single logical link. (Upper-layer software sees this link aggregation group as a single logical interface.) The single logical link can carry traffic at higher data rates than a single interface because the traffic is distributed across all of the physical ports that make up the link aggregation group.

Using link aggregation provides the following capabilities:

- Increased network bandwidth — The increase is incremental based on the number and type of ports, or NICs, added to the link aggregation group.
- Fault tolerance — If a port in a link aggregation group fails, the software detects the failure and reroutes traffic to the other available ports. This capability is available for DEGPA (alt) and DE60x (ee) devices only.
- Load sharing — A link aggregation group performs load sharing of both inbound and outbound traffic. When transmitting packets, the system uses a load distribution algorithm to determine on which attached port to transmit the packets.

You can use a link aggregation group virtual interface for the following point-to-point connections: server-to-server and server-to-switch. For more information see the *Network Administration: Connections* guide.

1.5 File System Resources

File-system tuning is important for the Advanced File System (AdvFS) and the Network File System (NFS). In general, file-system tuning will improve the performance of I/O-intensive user applications. The following sections discuss the file system resources for AdvFS, UNIX File System (UFS), and NFS.

1.5.1 Using AdvFS

The Advanced File System (AdvFS) file system differs from the traditional UNIX File System (UFS). With AdvFS you can modify your system

configuration at any time without shutting down the system. Because AdvFS with AdvFS utilities supports a multivolume file system, you can easily add or remove storage as your system requirements change. In addition, Logical Storage Manager (LSM) volumes and storage area networks (SANs) can be used for AdvFS storage.

In contrast, the UFS model is rigid. Each disk (or disk partition) contains a single file system. The directory hierarchy layer of UFS is bound tightly to the physical storage layer. When a file system becomes full, this tight binding makes it impossible to move selected files onto another disk without changing the full pathnames of those files. The task of dividing a logical directory into directory subtrees and mapping the subtrees onto separate disks requires careful consideration. Even with extensive planning, adjustments to the directory structure are limited with the UFS model.

1.5.1.1 Using the UBC

Caching improves performance when data is reused frequently. AdvFS uses a dynamic memory cache called the **Unified Buffer Cache (UBC)** to manage file metadata and user data.

By using the UBC for caching, AdvFS can maintain file data in memory as long as memory is available. If other system resources require some of the memory in use by the file system cache, the UBC can reclaim some of the memory used by the file system and reissue the needed memory to the resource requiring it.

Because AdvFS uses the UBC to control caching, the cache is tuned with the UBC tunable parameters. These include:

- Variables that modify the maximum percentage of physical memory that the UBC can use at one time.
- The percentage of pages that must be dirty before the UBC starts writing them to disk.
- The maximum amount of memory allocated to the UBC that can be used to cache a single file.

See Chapter 11 for the guidelines to modify these parameters.

1.5.2 Using NFS

The network file system (NFS) allows users to access files transparently across networks. The NFS supports a spectrum of network topologies, from small and simple networks to large and complex networks. The NFS shares the Unified Buffer Cache (UBC) with the virtual memory subsystem and local file systems.

File-system tuning is important for NFS because processing NFS requests consumes the majority of CPU and wall clock time. Ideally, the UBC hit rate should be high. Increasing the UBC hit rate can require additional memory or a reduction in the size of other file-system caches.

NFS uses a simple stateless protocol, which requires that each client request be complete and self-contained and that the server completely process each request before sending an acknowledgment back to the client.

Improving performance on a system that is used only for serving NFS differs from tuning a system that is used for general timesharing, because an NFS server runs only a few small user-level programs, which consume few system resources. There is minimal paging and swapping activity, so memory resources should be focused on caching file system data.

See Chapter 5 and Chapter 10 for more information on NFS tuning.

1.6 Memory Resources

Sufficient memory resources are vital to system performance. Configurations running CPU and memory-intensive applications often require very-large memory (**VLM**) systems that utilize 64-bit architecture, multiprocessing, and at least 2 GB of memory. Very-large database (**VLDB**) systems are VLM systems that also utilize complex storage configurations.

The total amount of **physical memory** is determined by the capacity of the memory boards installed in your system. The **virtual memory (vm) subsystem** tracks and manages this memory in 8-KB portions called **pages**, distributing them among the following areas:

- **Static wired memory**

  Allocated at boot time and used for operating system data and text and for system tables, static wired memory is also used by the metadata buffer cache, which holds recently accessed UNIX File System (UFS) and CD-ROM File System (CDFS) metadata.

- **Dynamically wired memory**

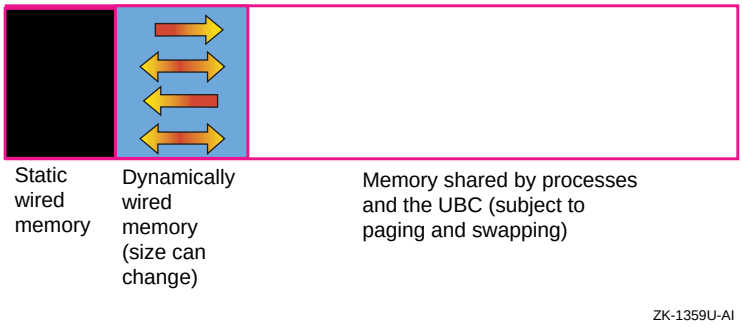
  Dynamically wired memory is used for dynamically allocated data structures, such as system hash tables. User processes also allocate dynamically wired memory for address space by using virtual memory locking interfaces, including the `mlock` function. The amount of dynamically wired memory varies according to the demand. The `vm` subsystem attribute `vm_syswiredpercent` specifies the maximum amount of memory that a user process can wire (by default, this is 80 percent of physical memory).

- **Physical memory for processes and data caching**

Physical memory that is not **wired** is referred to as **pageable memory**. It is used for processes' most-recently accessed **anonymous memory** (modifiable virtual address space) and **file-backed memory** (memory that is used for program text or shared libraries). Pageable memory is also used to cache the most-recently accessed UFS file system data for reads and writes and for page faults from mapped file regions, in addition to AdvFS metadata and file data. The virtual memory subsystem allocates physical pages according to the process and file system demand.

Figure 1–9 shows the division of physical memory.

Figure 1–9: Physical Memory Usage



1.6.1 Paging and Swapping

Physical memory is a resource that all active processes use. Often there is not enough physical memory to accommodate all active processes on the system. To provide more physical memory, the `vm` subsystem monitors the amount of available physical memory and might transfer pages to a secondary memory device called a **swap device**. A swap device is a block device in a configured section of a disk. The kernel retrieves pages from a swap device on demand when a process references the pages. This memory management policy is called **paging**.

Under heavy loads, an entire process might be transferred to a swap device. A process called the swapper manages the transfer of pages between physical memory and a swap device. This memory management policy is called **swapping**.

See Chapter 12 for more information on how to tune attributes that relate to paging and swapping.

1.6.2 Caching Data

The kernel caches (temporarily stores) in memory recently accessed data. Caching data is effective because data is frequently reused and it is much

faster to retrieve data from memory than from disk. When the kernel requires data, it checks if the data was cached. If the data was cached, it is returned immediately. If the data was not cached, it is retrieved from disk and cached. File system performance is improved if the cached data is later reused.

Cached data can be information about a file, user or application data, or metadata, which is data that describes an object for example, a file. The following list identifies the types of data that are cached:

- A file name and its corresponding `vnode` is cached in the namei cache (Section 11.1.2).
- UFS user and application data and AdvFS user and application data and metadata are cached in the Unified Buffer Cache (UBC) (Section 11.1.3).
- UFS metadata is cached in the metadata buffer cache (Section 11.1.4).
- AdvFS open file information is cached in access structures (Section 11.1.5).

1.7 CPU Resources

CPUs support different processor speeds and onboard **cache** sizes. In addition, you can choose single-CPU systems or **multiprocessor** systems, which allow two or more processors to share common physical memory. Environments that are CPU-intensive, such as large database environments, require multiprocessing systems to handle the workload.

An example of a multiprocessing system is a symmetrical multiprocessing (**SMP**) system, in which the CPUs execute the same version of the operating system, access common memory, and execute instructions simultaneously.

When programs are executed, the operating system moves data and instructions through CPU caches, physical memory, and disk swap space. Accessing the data and instructions occurs at different speeds, depending on the location. Table 1–6 describes the various hardware resources.

Table 1–6: Memory Management Hardware Resources

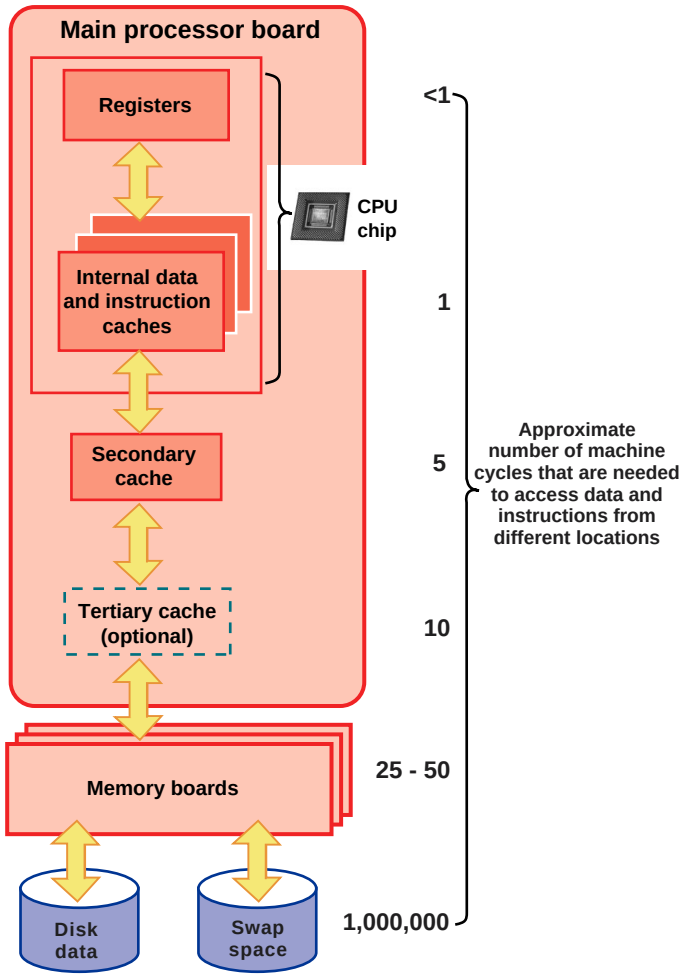
Resource	Description
CPU chip caches	Various internal caches reside in the CPU chip. They vary in size, up to a maximum of 64 KB, depending on the processor. These caches include the translation look aside buffer, the high-speed internal virtual-to-physical translation cache, the high-speed internal instruction cache, and the high-speed internal data cache.
Secondary cache	The secondary direct-mapped physical data cache is external to the CPU, but usually resides on the main processor board. Block sizes for the secondary cache vary from 32 bytes to 256 bytes (depending on the type of processor). The size of the secondary cache ranges from 128 KB to 8 MB.
Tertiary cache	The tertiary cache is not available on all Alpha CPUs; otherwise, it is identical to the secondary cache.
Physical memory	The actual amount of physical memory varies.
Swap space	Swap space consists of one or more disks or disk partitions (block special devices).

The hardware logic and the **Privileged Architecture Library (PAL)** code control much of the movement of addresses and data among the CPU cache, the secondary and tertiary caches, and physical memory. This movement is transparent to the operating system.

Movement between caches and physical memory is significantly faster than movement between disk and physical memory, because of the relatively slow speed of disk I/O. Applications should utilize caches and avoid disk I/O operations whenever possible.

Figure 1–10 shows how instructions and data are moved among various hardware components during program execution, and shows the machine cycles needed to access data and instructions from the hardware locations.

Figure 1–10: Moving Instructions and Data Through the Memory Hardware



For more information on the CPU, secondary cache, and tertiary cache, see the *Alpha Architecture Reference Manual*.

There are several ways that you can optimize CPU performance. You can reschedule processes or use the Class Scheduler to allocate a percentage of CPU time to a task or application. This allows you to reserve a majority of CPU time for important processes, while limiting CPU usage by less critical processes. See Section 13.2.2 for more information.

1.8 Identifying a Resource Model for Your Workload

Before you can plan or tune a configuration, you must identify a resource model for your workload. That is, you must determine if your applications are memory-intensive or CPU-intensive, and how they perform disk and network I/O. This information will help you to choose the configuration and tuning guidelines that are appropriate for your workload.

For example, if a database server performs large sequential data transfers, choose a configuration that provides high bandwidth. If an application performs many disk write operations, you may not want to choose a RAID1 (mirrored) configuration.

Use Table 1–7 to help you determine the resource model for your workload and identify a possible configuration solution for each model.

Table 1–7: Resource Models and Possible Configuration Solutions

Resource Model	Configuration Solution
CPU-intensive	Multiprocessing system, fast CPUs, or hardware RAID subsystem
Memory-intensive	VLM system or large onboard CPU cache
Requires large amount of disk storage	System with a large I/O capacity, LSM, or hardware RAID subsystem
Requires low disk latency	Solid-state disks, fast disks, RAID array, or Fibre Channel
Requires high throughput	Solid-state disks, high-performance SCSI adapters, striping, RAID5, or dynamic parity RAID (adaptive RAID3/5)
Requires high bandwidth	Solid-state disks, high-performance adapters, wide devices, RAID3, or dynamic parity RAID
Performs many large sequential data transfers	High-performance disks, wide devices, striping, parity RAID
Performs many small data transfers	RAID5
Issues predominantly read transfers	Mirroring, RAID5, or striping
Issues predominantly write transfers	Prestoserve or write-back cache
Performs many network operations	Multiple network adapters, NetRAIN, or high-performance adapters
Application must be highly available	Cluster
Data must be highly available	Mirroring (especially across different buses) or parity RAID
Network I/O-intensive	Multiple network adapters or NetRAIN

1.9 Most Commonly Tuned Subsystems

This manual describes how to tune many subsystem attributes. We recommend tuning only those attributes that are specific to your system and performance problem. The five most commonly tuned subsystems are:

- Virtual Memory (`vm`)

  `new_wire_method` (Section 4.4.1.1)
  `rad_gh_regions` (Section 4.4.1.2)
  `gh_chunks` (Section 4.4.1.2.2)
  `ubc_maxpercent` (Section 4.4.1.3)
  `ubc_borrowpercent` (Section 4.4.1.4)
  `vm_ubcseqstartpercent` (Section 4.4.1.6)
  `vm_ubcdirtypercent` (Section 4.4.1.7)
  `vm_swap_eager` (Section 4.4.1.8)

- Interprocess Communication (`ipc`)

  `ssm_threshold` (Section 4.4.4.1)
  `shm_max` (Section 4.4.4.2)
  `shm_min` (Section 4.4.4.3)
  `shm_mni` (Section 4.4.4.4)
  `shm_seg` (Section 4.4.4.5)

- Process (`proc`)

  `per_proc_stack_size` (Section 4.4.6.1)
  `max_per_proc_stack_size` (Section 4.4.6.2)
  `per_proc_data_size` (Section 4.4.6.3)
  `max_per_proc_data_size` (Section 4.4.6.4 and Section 6.2.2.4)
  `per_proc_address_space` (Section 4.4.6.5)
  `max_per_proc_address_space` (Section 4.4.6.6 and Section 6.2.2.5)
  `max_proc_per_user` (Section 4.4.6.7 and Section 6.2.2.2)
  `max_threads_per_user` (Section 4.4.6.8 and Section 6.2.2.3)
  `maxusers` (Section 4.4.6.9 and Section 6.2.2.1)

- Internet (`inet`)

  `udp_sendspace` (Section 4.4.5.1)
  `udp_recvspace` (Section 4.4.5.2)
  `udp_unserreserved` (Section 4.4.5.3)
  `tcbhashsize` (Section 6.2.1.1)
  `pmtu_enabled` (Section 6.2.1.2)
  `ipport_userreserved` (Section 6.2.1.3)

- Socket (`socket`)

    `somaxconn` (Section 6.2.3.1)
    `sominconn` (Section 6.2.3.2)
    `sbcompress-threshold` (Section 6.2.3.3)

This manual describes how to tune your system by application type and component. Before tuning your system, you need to understand your system hardware configuration (see Section 1.1 for more information). The most commonly tuned subsystems are mentioned throughout this manual, but only tune those attributes that are related to your performance problem.

The following chapters describe which attributes to tune for improving system performance:

- Tuning by Application Type (Part 2):

    Tuning Oracle (Chapter 4)
    Tuning Network File Systems (Chapter 5)
    Tuning Internet Servers (Chapter 6)

- Tuning by Component (Part 3):

    Managing System Resource Allocation (Chapter 8)
    Managing Disk Storage Performance (Chapter 9)
    Managing Network Performance (Chapter 10)
    Managing File System Performance (Chapter 11)
    Managing Memory Performance (Chapter 12)
    Managing CPU Performance (Chapter 13)

For more information on subystem attributes, see `sys_attrs`(5).

2

Gathering System and Performance Information

You must gather a wide variety of performance information to identify performance problems or areas where performance is deficient.

Some symptoms or indications of performance problems are obvious. For example, applications complete slowly or messages appear on the console, indicating that the system is out of resources. Other problems or performance deficiencies are not obvious and can be detected only by monitoring system performance.

There are various commands and utilities that you can use to gather system performance information. It is important that you gather statistics under a variety of conditions. Comparing sets of data will help you diagnose performance problems.

For example, to determine how an application affects system performance, you can gather performance statistics without the application running, start the application, and then gather the same statistics. Comparing different sets of data will enable you to identify whether the application is consuming memory, CPU, or disk I/O resources.

In addition, you must gather information at different stages during the application processing to obtain accurate performance information. For example, an application may be I/O-intensive during one stage and CPU-intensive during another.

This chapter describes how to perform the following tasks:

- Using a methodology approach to solve performance problems (Section 2.1)
- Obtaining information about system events (Section 2.2)
- Using the primary tools for gathering information (Section 2.3)
- Using secondary tools to gather information (Section 2.4)
- Continuously monitoring performance (Section 2.5)

After you identify a performance problem or an area in which performance is deficient, you can identify an appropriate solution. See Part 2 for

information about tuning by application, and see Part 3 for information about tuning by component to improve system performance.

2.1 Methodology Approach to Solving Performance Problems

There are five recommended steps to diagnose a performance problem. Before you begin, you must become familiar with the terminology and concepts relating to performance and availability. See Chapter 1 for more information.

In addition, you must understand how your application utilizes system resources, because not all configurations and tuning guidelines are appropriate for all types of workloads. For example, you must determine if your applications are memory-intensive or CPU-intensive, or if they perform many disk or network operations. See Section 1.8 for information about identifying a resource model for your configuration.

To diagnose performance problems, follow these steps:

1. Before you begin, you must understand your system hardware configuration. To identify and manage your hardware components use the`hwmgr` utility (see Section 1.1 or Section 2.3.1 for more information).
2. Run the `sys_check` utility, but before you do so perform an analysis of the operating system parameters and kernel attributes that tune the performance of your system. This tool can be used to diagnose performance problems. See Section 2.3.3 for more information.
3. Verify and know your software configuration errors. You can use `sys_check` to diagnose performance problems. See Section 2.2 for more information about obtaining information for system events.
4. Determine what type of application you are using and categorize your application as an Oracle, Network File System, or internet server application. If you are tuning your system by applications, see the following chapters:
   - Tuning Oracle (Chapter 4)
   - Tuning Network File Systems (Chapter 5)
   - Tuning Internet Servers (Chapter 6)
5. Find the bottleneck or the system resource that is causing a performance degradation. Determine the performance problem by plotting the following information:
   - CPU — Idle system time and user time

- Memory — Sum of active or inactive pages that are being used by processes, UBC and wired memory.
- Disk I/O — Transactions per second and blocks per second

Use the `collect` command to gather performance data while the system is under load or manifesting the performance problem. After you gather the performance information, use the `collgui` graphical interface to plot the data. For information on how to use `collgui`, see Section 2.3.2.2. For more information about identifying a resource model for your workload, see Section 1.8.

2.2 Obtaining Information About System Events

Set up a routine to continuously monitor system events that will alert you when serious problems occur. Periodically examining event and log files allows you to correct a problem before it affects performance or availability, and helps you diagnose performance problems.

The system event logging facility and the binary event logging facility log system events. The system event logging facility uses the `syslog` function to log events in ASCII format. The `syslogd` daemon collects the messages logged from the various kernel, command, utility, and application programs. This daemon then writes the messages to a local file or forwards the messages to a remote system, as specified in the `/etc/syslog.conf` event logging configuration file. Periodically monitor these ASCII log files for performance information.

The binary event logging facility detects hardware and software events in the kernel and logs detailed information in binary format records. The binary event logging facility uses the `binlogd` daemon to collect various event log records. The daemon then writes these records to a local file or forwards the records to a remote system, as specified in the `/etc/binlog.conf` default configuration file.

You can examine the binary event log files by using the following methods:

- The Event Manager (EVM) uses the binary log files to communicate event information to interested parties for immediate or later action. See Section 2.2.1 for more information about EVM.
- DECevent is a rules-based translation and reporting utility that provides event translation for binary error log events. EVM uses DECevent's translation facility, `dia`, to translate binary error log events into human-readable form. Compaq Analyze performs a similar role on some EV6 series processors.

  For more information about DECevent, see Section 2.2.2 or `dia`(8).

  For more information on Compaq Analyze, see Section 2.2.3 or `ca`(8).

In addition, we recommend that you configure crash dump support into the system. Significant performance problems may cause the system to crash, and crash dump analysis tools can help you diagnose performance problems.

See the *System Administration* manual for more information about event logging and crash dumps.

2.2.1 Using Event Manager

Event Manager (EVM) allows you to obtain event information and communicate this information to interested parties for immediate or later action. Event Manager provides the following features:

- Enables kernel-level and user-level processes and components to post events.
- Enables event consumers, such as programs and users, to subscribe for notification when selected events occur.
- Supports existing event channels such as the binary logger daemon.
- Provides a graphical user interface (GUI) that enables users to review events.
- Provides an application programming interface (API) library that enables programmers to write routines that post or subscribe to events.
- Supports command-line utilities for administrators to configure and manage the EVM environment and for users to post or retrieve events.

See the *System Administration* manual for more information about EVM.

2.2.2 Using DECevent

The DECevent utility continuously monitors system events through the binary event logging facility, decodes events, and tracks the number and the severity of events logged by system devices. DECevent analyzes system events, attempts to isolate failing device components, and provides a notification mechanism (for example, mail) that can warn of potential problems.

You must register a license to use DECevent's analysis and notification features, or these features may also be available as part of your service agreement. A license is not needed to use DECevent to translate the binary log file to ASCII format.

See the *DECevent Translation and Reporting Utility* manual for more information.

2.2.3 Using Compaq Analyze

Compaq Analyze is a fault analysis utility designed to provide analysis for single error/fault events, and multiple event and complex analysis. Compaq Analyze provides system analysis that uses other error/fault data sources in addition to the traditional binary error log.

Compaq Analyze provides background automatic analysis by monitoring the active error log and processing events as they occur. The events in the error log file are checked against the analysis rules. If one or more of the events in the error log file meets the conditions specified in the rules, the analysis engine collects the error data and creates a problem report containing a description of the problem and any corrective actions required. Once the problem report is created, it is distributed in accordance with your notification preferences.

Note that recent Alpha EV6 processors are supported only by Compaq Analyze and not DECevent.

You can download the latest version of Compaq Analyze and other Web Based Enterprise Service Suite (WEBES) tools and documentation from the following location:

http://www.compaq.com/support/svctools/webes

Download the kit from the Web site, saving it to `/var/tmp/webes`. Unpack the kit using a command similar to the following:

```
# tar -xvf <tar file name>
```

Use the following command to install the Compaq Web Based Enterprise Service Suite:

```
# setld -l /var/temp/webes/kit
```

During the installation, you can safely select the default options. However, you might not want to install all the optional WEBES tools. Only Compaq Analyze is used by EVM. See the separate Compaq Analyze documentation and `ca`(8) for more information.

2.2.4 Using System Accounting and Disk Quotas

Set up system accounting, which allows you to obtain information about the resources consumed by each user. Accounting can track the amount of CPU usage and connect time, the number of processes spawned, memory and disk usage, the number of I/O operations, and the number of print operations.

You should establish Advanced File System (AdvFS) and UNIX file system (UFS) **disk quotas** to track and control disk usage. Disk quotas allow you to limit the disk space available to users and to monitor disk space usage.

See the *System Administration* manual for information about system accounting and UFS disk quotas. See the *AdvFS Administration* manual for information about AdvFS quotas.

2.3 Primary Tools for Gathering Information

The following utilities are the primary tools for gathering performance information:

- `hwmgr` utility (Section 2.3.1)
- `collect` utility (Section 2.3.2)
- `sys_check` utility (Section 2.3.3)

2.3.1 Gathering Hardware Information Using the hwmgr Utility

The principal command that you use to manage hardware is the `hwmgr` command-line interface (CLI). Other interfaces, such as the SysMan tasks, provide a limited subset of the features provided by `hwmgr`. Using the `hwmgr` command enables you to connect to an unfamiliar system, obtain information about its component hierarchy and allows you to set attributes for specific components.

Use the `view` command to view the hierarchy of hardware within a system. This command enables you to find what adapters are controlling devices, and discover where adapters are installed on buses. The following example shows the hardware component hierarchy on a small system that is not part of a cluster:

```
# hwmgr view hierarchy

HWID: Hardware component hierarchy
-------------------------------------------
 1: platform AlphaServer 800 5/500
 2:   cpu CPU0
 4:   bus pci0
 5:     scsi_adapter isp0
 6: scsi_bus scsi0
18:   disk bus-0-targ-0-lun-0 dsk0
19:   disk bus-0-targ-4-lun-0 cdrom0
20:     graphics_controller trio0
 8:     bus eisa0
 9: serial_port tty00
10: serial_port tty01
11: parallel_port lp0
12: keyboard PCXAL
13: pointer PCXAS
14: fdi_controller fdi0
15:   disk fdi0-unit-0 floppy0
16:     network tu0
17:     network tu1
output truncated
```

Some components might appear as multiple entries in the hierarchy. For example, if a disk is on a SCSI bus that is shared between two adapters, the hierarchy shows two entries for the same device. You can obtain similar views of the system hardware hierarchy by using the SysMan Station GUI. See the *System Administration* manual for information on running the SysMan Menu. Section 13.2.2.6 describes how to use the graphical interface. See the online help for more information on valid data entries.

To view a specific component in the hierarchy, use the `grep` command. The following example shows output for the CPU hardware component:

```
# hwmgr view hierarchy | grep "cpu"
 2:       cpu qbb-0 CPU0
 3:       cpu qbb-0 CPU1
 4:       cpu qbb-0 CPU2
 5:       cpu qbb-0 CPU3
 7:       cpu qbb-1 CPU5
 8:       cpu qbb-1 CPU6
 9:       cpu qbb-1 CPU7
10:       cpu qbb-2 CPU8
11:       cpu qbb-2 CPU9
12:       cpu qbb-2 CPU10
13:       cpu qbb-2 CPU11
```

The `display hierarchy` command displays the currently registered hardware components which have been placed in the system hierarchy. Components that have a flagged status are identified in the command output with the following codes:

- (`!`) warning
- (`X`) critical
- (`-`) inactive

See `hwmgr`(8) for an explanation of these codes.

To view all of the SCSI devices attached to the system (disks and tapes), use the following command:

```
# hwmgr show scsi
```

To view how many RAID array controllers can be seen from the host use the following command:

```
# hwmgr show scsi | grep scp

        SCSI                DEVICE   DEVICE  DRIVER NUM  DEVICE FIRST
 HWID:  DEVICEID HOSTNAME   TYPE     SUBTYPE OWNER  PATH FILE   VALID PATH
------------------------------------------------------------------------
266:    30      wf99    disk    none    0   20   scp0  [2/0/7]
274:    38      wf99    disk    none    0   20   scp1  [2/1/7]
282:    46      wf99    disk    none    0   20   scp2  [2/2/7]
290:    54      wf99    disk    none    0   20   scp3  [2/3/7]
298:    62      wf99    disk    none    0   20   scp4  [2/4/7]
306:    70      wf99    disk    none    0   20   scp5  [2/5/7]
314:    78      wf99    disk    none    0   20   scp6  [2/6/7]
322:    86      wf99    disk    none    0   20   scp7  [2/7/7]
330:    94      wf99    disk    none    0   20   scp8  [2/8/7]
338:    102     wf99    disk    none    0   20   scp9  [2/9/7]
```

```
346:    110    wf99    disk    none    0   20   scp10 [2/10/7]
354:    118    wf99    disk    none    0   20   scp11 [2/11/7]
```

The scp in the previous example represents the service control port and is the address that a RAID array (HSG) presents itself for administrative and diagnostic purposes.

For more information about the hwmgr command, see the *Hardware Management* manual or hwmgr(8).

2.3.2 Gathering System Information by Using the collect Utility

The collect utility is a system monitoring tool that records or displays specific operating system data. It also gathers the vital system performance information for specific subsystems, such as file systems, memory, disk, process data, CPU, network, message queue, LSM, and others. The collect utility creates minimal system overhead and is highly reliable. It also provides extensive and flexible switches to control data collection and playback. You can display data at the terminal, or store it in either a compressed or uncompressed data file. Data files can be read and manipulated from the command line.

To ensure that the collect utility delivers reliable statistics, it locks itself into memory using the page-locking function plock( ), and by default cannot be swapped out by the system. It also raises its priority using the priority function nice( ). However, these measures should not have any impact on a system under normal load, and they should have only a minimal impact on a system under extremely high load.

You can invoke the collect utility from the collgui graphical user interface or from the command line. If you are using the graphic user interface, run cfilt on the command line to filter collect's data used by collgui and user scripts. For more information see collect(8).

The following example shows how to run a full data collection and display the output at the terminal using the standard interval of 10 seconds:

```
# /usr/sbin/collect
```

This command is similar to the output monitoring commands such as vmstat(1), iostat(1), netstat(1), and volstat(8).

Use the -s option to select subsystems for inclusion in the data collection, or use the -e (exclude) option to exclude subsystems from the data collection.

The following output specifies only data from the file system subsystem:

```
# /usr/sbin/collect -sf
# FileSystem Statistics
# FS    Filesystem         Capacity    Free
0       root_domain#root     128        30
```

```
1       usr_domain#usr       700        147
3       usr_domain#var       700        147
```

The option letters map to the following subsystems:

- *p* — Specifies the process data
- *m* — Specifies the memory data
- *d* — Specifies the disk data
- *l* — Specifies the LSM volume data
- *n* — Specifies the network data
- *c* — Specifies the CPU data
- *f* — Specifies the file system data
- *tyy* — Specifies the terminal data

When you are collecting process data, use the `-S` (sort) and `-n` *X* (number) options to sort data by percentage of CPU usage and to save only *X* processes. Target-specific processes using the *Plist* option, where *list* is a list of process identifiers, comma-separated with blanks.

If there are many (greater than 100) disks connected to the system being monitored, use the `-D` option to monitor a particular set of disks.

Use the `collect` utility with the `-p` option to read multiple binary data files and play them back as one stream, with monotonically increasing sample numbers. You can also combine multiple binary input files into one binary output file, using the `-p` option with the input files and the `-f` option with the output file.

The `collect` utility will combine input files in whatever order you specify on the command line. This means that the input files must be in strict chronological order if you want to do further processing of the combined output file. You can also combine binary input files from different systems, made at different times, with differing subsets of subsystems for which data has been collected. Filtering options such as `-e`, `-s`, `-P`, and `-D` can be used with this utility.

See `collect`(8) for more information.

2.3.2.1 Configuring collect to Automatically Start on System Reboot

You can configure `collect` to automatically start when the system reboots. This is particularly useful for continuous monitoring of subsystems and processes. It is essential for diagnosing problems and performance issues. On each system, use the `rcmgr` command with the `set` operation to configure the following values in the `/etc/rc.config*` file. For example:

```
% rcmgr set COLLECT_AUTORUN 1
```

A value of 1 sets collect to automatically start when the system reboots. A value of 0 (the default) causes collect to not start on reboot:

```
% rcmgr set COLLECT_ARGS " -ol -i 10:60 \ -f
/var/adm/collect.dated/collect -H d0:5,1w "
```

A null value causes collect to start with the following default values:

```
-i60, 120 -f /var/adm/collect.dated -W 1h -M 10, 15
```

Direct output from collect should be written to a local file system, not a NFS-mounted file system, to prevent important diagnostic data from being lost during system or network problems and to prevent any consequent system problems arising from collect output being blocked by a nonresponsive file system.

See rcmgr(8) for more information.

2.3.2.2 Plotting collect Datafiles

Use either collgui (a graphical interface for the collect command) or cflit (a filter for the collect command) to export collect datafiles to Excel.

Note

To run collgui, you need Perl and Perl/TK. They are freely downloadable from the collect FTP site: **ftp://ftp.digital.com/pub/DEC/collect**

To plot information using the collgui graphical interface, follow these steps:

1. Run collgui in debug mode:

   ```
   >> collgui -d "collect datafile"
   ```

2. Select the desired subsystem and click on Display.
3. Return to the shell that collgui was started in. You will see that collgui has created the /var/tmp directory. The file name is collgui.xxxx, where xxxx are integers. The data file (collgui.xxxx) is exportable to Excel. Copy it to a Windows system.
4. On your Windows system, start Excel and open collgui.xxxx. You might have to change files of Type: field to “All files (*.*)”.
5. In Excel 2000, a text import wizard will pop up.
6. In Excel 2000, select data type *Delimited*, then select *Next*.
7. In Excel 2000, select Tabs and Spaces as *Delimiters*, then select *Next*.

8. In the Data Preview pane, select the columns that you want to import using the *Shift* key select *Finish*.
9. You should now see the columns displayed in your worksheet.

To plot information using the `cflit` collect filter, follow these steps:

1. Use `cfilt` to generate the data file. For example, if you choose to display the `system time`, `physical memory` used for the process data, and `user+system time`, wait time for the Single CPU field, enter the following command:

```
cfilt -f "collect datafile" 'sin:WAIT:USER+SYS' 'pro:Systim#:RSS#'
> /var/tmp/collgui.xxxx
```

   Copy the `collgui` data file to your Windows system.
2. Follow steps 4–9 in the `collgui` previous procedure.

For more information, see the following Web site:
`http://www.tru64unix.compaq.com/collect/collect_faq.html`

2.3.3 Checking the Configuration by Using the sys_check Utility

The `sys_check` utility performs an analysis of operating system parameters and kernel attributes that tune the performance of your system. The utility checks memory and CPU resources, provides performance data and lock statistics for SMP systems and for kernel profiles, and outputs any warnings and tuning guidelines.

The `sys_check` utility creates an HTML file that describes the system configuration, and can be used to diagnose problems. The report generated by `sys_check` provides warnings if it detects problems with any current settings. Use `sys_check` utility in conjunction with the event management and system monitoring tools to provide a complete overview and control of system status.

Consider applying the `sys_check` utility's configuration and tuning guidelines before applying any advanced tuning guidelines.

Note

You may experience impaired system performance while running the `sys_check` utility. Invoke the utility during offpeak hours to minimize the performance impact.

You can invoke the `sys_check` utility from the SysMan graphical user interface or from the command line. If you specify `sys_check` without any command-line options, it performs a basic system analysis and creates an

HTML file with configuration and tuning guidelines. Options that you can specify at the command line include:

- The `-all` option provides information about all subsystems, including security information and `setld` inventory verification.
- The `-perf` option provides only performance data and excludes configuration data. This may take 5 to 10 minutes to complete.
- The `-escalate` option creates escalation files required for reporting problems.

See `sys_check`(8) for more information.

2.4 Secondary Tools for Gathering Information

The following utilities are the secondary tools used to gather performance information:

Gathering system information:

- `lockinfo` utility (Section 2.4.1)
- `sched_stat` utility (Section 2.4.2)

Gathering network information:

- `nfsstat` utility (Section 2.4.3)
- `tcpdump` utility (Section 2.4.4)
- `netstat` command (Section 2.4.5)
- `ps axlmp` command (Section 2.4.6)
- `nfsiod` daemon (Section 2.4.7)
- `nfswatch` command (Section 2.4.8)

2.4.1 Gathering Locking Statistics by Using the lockinfo Utility

The `lockinfo` utility collects and displays locking statistics for the kernel SMP locks. It uses the `/dev/lockdev` pseudodriver to collect data. Locking statistics can be gathered when the `lockmode` attribute for the `generic` subsystem is set to 2 (the default), 3, or 4.

To gather statistics with `lockinfo`, follow these steps:

1. Start up a system workload and wait for it to get to a steady state.
2. Start `lockinfo` with `sleep` as the specified command and some number of seconds as the specified *`cmd_args`*. This causes `lockinfo` to gather statistics for the length of time it takes the `sleep` command to execute.

3. Based on the first set of results, use `lockinfo` again to request more specific information about any lock class that shows results, such as a large percentage of misses, which is likely to cause a system performance problem.

The following example shows how to gather locking statistics for each processor over a period of 60 seconds:

```
# lockinfo -percpu sleep 60
hostname:       sysname.node.corp.com
lockmode:       4  (SMP DEBUG with  kernel mode preemption enabled)
processors:     4
start time:     Wed Jun  9 14:45:08 1999
end time:       Wed Jun  9 14:46:08 1999
command:        sleep 60

          tries    reads    trmax    misses   percent  sleeps    waitmax  waitsum
                                               misses             seconds  seconds
bsBuf.bufLock  (S)
 0      1400786        0    45745     47030       3.4       0    0.00007  0.15526
 1      1415828        0    45367     47538       3.4       0    0.00006  0.15732
 2      1399462        0    33076     48507       3.5       0    0.00005  0.15907
 3      1398336        0    31753     48867       3.5       0    0.00005  0.15934

        ------------------------------------------------------------------------

ALL     5614412        0    45745    191942       3.4       0    0.00007  0.63099

lock.l_lock  (S)
 0      1360769        0    40985     18460       1.4       0    0.00005  0.04041
 1      1375384        0    20720     18581       1.4       0    0.00005  0.04124
 2      1375122        0    20657     18831       1.4       0    0.00009  0.04198

         -----------------------------------------------------------------------

ALL     5483049        0    40985     74688       1.4       0    0.00009  0.16526
...inifaddr_lock  (C)
 0            0        0        1         0       0.0       0    0.00000  0.00000
 1            1        1        1         0       0.0       0    0.00000  0.00000
 2            0        0        1         0       0.0       0    0.00000  0.00000
 3            0        0        1         0       0.0       0    0.00000  0.00000

         -----------------------------------------------------------------------

ALL           1        1        1         0       0.0       0    0.00000  0.00000

total simple_locks = 28100338                          percent unknown = 0.0
total rws_locks = 1466          percent reads = 100.0
total complex_locks = 2716146   percent reads = 33.2    percent unknown = 0.0
```

A locking problem is simply an indication that there is high contention for a certain type of resource. If contention exists for a lock related to I/O, and a particular application is spawning many processes that compete for the same files and directories, application or database storage design adjustments might be in order.

Applications that use System V semaphores can sometimes encounter locking contention if they create a very large number of semaphores in a single semaphore set because the kernel uses locks on each set of

semaphores. In this case, performance improvements might be realized by changing the application to use more semaphore sets, each with a smaller number of semaphores.

See `lockinfo`(8) for more information.

2.4.2 Gathering CPU Usage and Process Statistics by Using the sched_stat Utility

The `sched_stat` utility helps determine how well the system load is distributed among CPUs, what kinds of jobs are getting (or not getting) enough cycles on each CPU, and how well cache affinity is being maintained for these jobs. The `sched_stat` displays CPU usage and process-scheduling for SMP and NUMA platforms.

To gather statistics with `sched_stat`, follow these steps:

1. Start up a system workload and wait for it to get to a steady state.
2. Start `sched_stat` with `sleep` as the specified command and some number of seconds as the specified *`cmd_arg`*. This causes `sched_stat` to gather statistics for the length of time it takes the `sleep` command to execute.

For example, the following command causes `sched_stat` to collect statistics for 60 seconds and then print a report:

```
# /usr/sbin/sched_stat sleep 60
```

If you include options on the command line, only statistics for the specified options are reported. If you specify the command without any options, all options except for `-R` are assumed. See `sched_stat`(8) for more information.

2.4.3 Displaying Network and NFS Statistics by Using the nfsstat Utility

To display or reinitialize NFS and remote procedure call (RPC) statistics for clients and servers, including the number of packets that had to be retransmitted (`retrans`) and the number of times a reply transaction ID did not match the request transaction ID (`badxid`), enter:

```
# /usr/ucb/nfsstat
```

Information similar to the following is displayed:

```
Server rpc:
calls     badcalls  nullrecv   badlen   xdrcall
38903     0         0          0        0

Server nfs:
calls     badcalls
38903     0
```

```
Server nfs V2:
null      getattr   setattr    root      lookup     readlink   read
5  0%     3345  8%  61  0%     0  0%     5902 15%   250  0%    1497  3%
wrcache   write     create     remove    rename     link       symlink
0  0%     1400  3%  549  1%    1049  2% 352  0%    250  0%    250  0%
mkdir     rmdir     readdir    statfs
171  0%   172  0%   689  1%    1751  4%

Server nfs V3:
null      getattr   setattr    lookup    access    readlink   read
0  0%     1333  3%  1019  2%   5196 13%  238  0%   400  1%    2816  7%
write     create    mkdir      symlink   mknod     remove     rmdir
2560  6%  752  1%   140  0%    400  1%   0  0%     1352  3%   140  0%
rename    link      readdir    readdir+  fsstat    fsinfo     pathconf
200  0%   200  0%   936  2%    0  0%     3504  9%  3  0%      0  0%
commit
21  0%

Client rpc:
calls     badcalls  retrans    badxid    timeout   wait       newcred
27989     1         0          0         1         0          0
badverfs  timers
0         4

Client nfs:
calls     badcalls  nclget     nclsleep
27988     0         27988      0

Client nfs V2:
null      getattr   setattr    root      lookup    readlink   read
0  0%     3414 12%  61  0%     0  0%     5973 21%  257  0%    1503  5%
wrcache   write     create     remove    rename    link       symlink
0  0%     1400  5%  549  1%    1049  3%  352  1%   250  0%    250  0%
mkdir     rmdir     readdir    statfs
171  0%   171  0%   713  2%    1756  6%

Client nfs V3:
null      getattr   setattr    lookup    access    readlink   read
0  0%     666  2%   9  0%      2598  9%  137  0%   200  0%    1408  5%
write     create    mkdir      symlink   mknod     remove     rmdir
1280  4%  376  1%   70  0%     200  0%   0  0%     676  2%    70  0%
rename    link      readdir    readdir+  fsstat    fsinfo     pathconf
100  0%   100  0%   468  1%    0  0%     1750  6%  1  0%      0  0%
commit
10  0%
```

The ratio of timeouts to calls (which should not exceed 1 percent) is the most important thing to look for in the NFS statistics. A timeout-to-call ratio greater than 1 percent can have a significant negative impact on performance. See Chapter 10 for information on how to tune your system to avoid timeouts.

To display NFS and RPC information in intervals (seconds), enter:

```
# /usr/ucb/nfsstat -s -i number
```

The following example displays NFS and RPC information in 10–second intervals:

```
# /usr/ucb/nfsstat -s -i 10
```

If you are monitoring an experimental situation with nfsstat, reset the NFS counters to 0 before you begin the experiment. To reset counters to 0, enter:

```
# /usr/ucb/nfsstat -z
```

See nfsstat(8) for more information about command options and output.

2.4.4 Gathering Information by Using the tcpdump Utility

The tcpdump utility monitors and displays packet headers on a network interface. You can specify the interface on which to listen, the direction of the packet transfer, or the type of protocol traffic to display.

The tcpdump command allows you to monitor the network traffic associated with a particular network service and to identify the source of a packet. It lets you determine whether requests are being received or acknowledged, or in the case of slow network performance, to determine the source of network requests

Your kernel must be configured with the packetfilter option to use the command. For example:

```
# pfconfig +p +c tu0
```

The netstat -ni command displays the configured network interfaces. For example:

```
# netstat -ni
Name  Mtu  Network         Address          Ipkts Ierrs  Opkts Oerrs  Coll
tu0  1500  <Link>        00:00f8:22:f8:05     486404139    010939748   632583736
tu0  1500  16.140.48/24  16.140.48.156        486404139    010939748   632583736
tu0  1500  DLI           none                 486404139    010939748   632583736
sl0*  296  <Link>                               0   0       0  0  0
lo0  4096  <Link>                            1001631086 0 1001631086     0  0
lo0  4096  127/8         127.0.0.1           1001631086 0 1001631086     0  0
```

Use the netstat command output to determine which interface to use with the tcpdump command. For example:

```
# tcpdump -mvi tu0 -Nts1500
tcpdump: listening on tu0
Using kernel BPF filter

k-1.fc77a110 > foo.pmap-v2: 56 call getport prog "nfs"  V3  prot UDP  port 0 \
(ttl 30, id 20054)
foo.fc77a110 > k-1.pmap-v2: 28 reply getport 2049 (ttl 30, id 36169)
k-1.fd77a110 > foo.pmap-v2: 56 call getport prog "mount"  V3  prot UDP  port 0 \
(ttl 30, id 20057)
foo.fd77a110 > k-1.pmap-v2: 28 reply getport 1030 (ttl 30, id 36170)
k-1.fe77a110 > foo.mount-v3: 112 call mount "/pns2" (ttl 30, id 20062)
foo.fe77a110 > k-1.mount-v3: 68 reply mount  OSF/1 fh 19,17/1.4154.1027680688/4154.\
1027680688 (DF) (ttl 30, id 36171)
k-1.b81097eb > fubar.nfs-v3: 136 call fsinfo OSF/1 fh 19,17/1.4154.1027680688/4154.\
1027680688 (ttl 30, id 20067)
```

The `-s snaplen` option displays snaplen bytes of data from each packet rather than the default of 68. The default is adequate for IP, ICP, TCP, and UDP, but 500–1500 bytes is recommended for NFS and RPC adequate results.

See `tcpdump`(8) and `packetfilter`(7) for more information.

2.4.5 Monitoring Network Statistics by Using the netstat Command

To check network statistics, use the `netstat` command. Some problems to look for are:

- If the `netstat -i` command shows excessive amounts of input errors (`Ierrs`), output errors (`Oerrs`), or collisions (`Coll`), this may indicate a network problem; for example, cables are not connected properly or the Ethernet is saturated (see Section 2.4.5.1).
- Use the `netstat -is` command to check for network device driver errors (see Section 2.4.5.2).
- Use the `netstat -m` command to determine if the network is using an excessive amount of memory in proportion to the total amount of memory installed in the system.

  If the `netstat -m` command shows several requests for memory delayed or denied, this means that either physical memory was temporarily depleted or the kernel `malloc` free lists were empty (see Section 2.4.5.3).
- Each socket results in a network connection. If the system allocates an excessive number of sockets, use the `netstat -an` command to determine the state of your existing network connections (see Section 2.4.5.4).

  For Internet servers, the majority of connections usually are in a `TIME_WAIT` state.
- Use the `netstat -p ip` command to check for bad checksums, length problems, excessive redirects, and packets lost because of resource problems (see Section 2.4.5.5).
- Use the `netstat -p tcp` command to check for retransmissions, out of order packets, and bad checksums (see Section 2.4.5.6).
- Use the `netstat -p udp` command to check for bad checksums and full sockets (see Section 2.4.5.6).
- Use the `netstat -rs` command to obtain routing statistics (see Section 2.4.5.7).
- Use the `netstat -s` command to obtain display statistics related to the IP, ICMP, IGMP, TCP, and UDP protocol layers (see Section 2.4.5.8).

Most of the information provided by `netstat` is used to diagnose network hardware or software failures, not to identify tuning opportunities. See the *Network Administration: Connections* manual for more information on how to diagnose failures.

See `netstat`(1) for more information about the output produced by the various command options.

2.4.5.1 Input and Output Errors and Collisions

Network collisions are a normal part of network operations. A collision can occur when two or more Ethernet stations attempt to transmit simultaneously on the network. If a station is unable to access the network because another one is already using it, the station will stop trying to access the network for a short period of time, before attempting to access the network again. A collision occurs each time a station fails to access the network. Most network interface cards (NICs) will attempt to transmit a maximum of 15 times, after which they will drop the output packet and issue an `excessive collisions` error.

Use the output of the `netstat -i` command to check for input errors (`Ierrs`), output errors (`Oerrs`), and collisions (`Coll`). Compare the values in these fields with the total number of packets sent. High values may indicate a network problem. For example, cables may not be connected properly or the Ethernet may be saturated. A collision rate of up to 10 percent may not indicate a problem on a busy Ethernet. However, a collision rate of more than 20 percent could indicate a problem. For example:

```
# netstat -i
Name  Mtu  Network  Address             Ipkts Ierrs    Opkts  Oerrs  Coll
tu0   1500  Link    00:00:aa:11:0a:c1       0     0    43427  43427     0
tu0   1500  DLI     none                    0     0    43427  43427     0
tu1   1500  Link    bb:00:03:01:6c:4d  963447   138   902543  1118  80006
tu1   1500  DLI     none               963447   138   902543  1118  80006
tu1   1500  o-net   plume              963447   138   902543  1118  80006
.
.
.
```

2.4.5.2 Device Driver Errors

Use the `netstat -is` command to check for network device driver errors. For example:

```
# netstat -is
tu0 Ethernet counters at Tue Aug  3 13:57:35 2002
          191 seconds since last zeroed
     14624204 bytes received
      4749029 bytes sent
        34784 data blocks received
        11017 data blocks sent
      2197154 multicast bytes received
        17086 multicast blocks received
         1894 multicast bytes sent
```

```
 17 multicast blocks sent
932 blocks sent, initially deferred
347 blocks sent, single collision
666 blocks sent, multiple collisions
  0 send failures
  0 collision detect check failure
  1 receive failures, reasons include: Frame too long
  0 unrecognized frame destination
  0 data overruns
  0 system buffer unavailable
  0 user buffer unavailable
```

The previous example shows that the system sent 11,017 blocks. Of those blocks, 1,013 (347 + 666) blocks had collisions, which represents approximately 10 percent of the blocks sent. A collision rate of up to 10 percent may not indicate a problem on a busy Ethernet. However, a collision rate of more then 20 percent could indicate a problem.

In addition, the following fields should be 0 or a low single-digit number:

- send failures
- receive failures
- data overruns
- system buffer unavailable
- user buffer unavailable

2.4.5.3 Memory Usage

The `netstat -m` command shows statistics for network-related memory structures, including the memory that is being used for `mbuf` clusters. Use this command to determine if the network is using an excessive amount of memory in proportion to the total amount of memory installed in the system. If the `netstat -m` command shows several requests for memory (`mbuf`) clusters delayed or denied, this means that your system was temporarily short of physical memory. The following example is from a firewall server with 128 MB memory that does not have `mbuf` cluster compression enabled:

```
# netstat -m
 2521 Kbytes for small data mbufs (peak usage 9462 Kbytes)
78262 Kbytes for mbuf clusters (peak usage 97924 Kbytes)
 8730 Kbytes for sockets (peak usage 14120 Kbytes)
 9202 Kbytes for protocol control blocks (peak usage 14551
    2 Kbytes for routing table (peak usage 2 Kbytes)
    2 Kbytes for socket names (peak usage 4 Kbytes)
    4 Kbytes for packet headers (peak usage 32 Kbytes)
39773 requests for mbufs denied
    0 calls to protocol drain routines
98727 Kbytes allocated to network
```

The previous example shows that 39,773 requests for memory were denied. This indicates a problem because this value should be 0. The example also shows that 78 MB of memory has been assigned to `mbuf` clusters, and that 98 MB of memory is being consumed by the network subsystem.

If you increase the value of the socket subsystem attribute sbcompress_threshold to 600, the memory allocated to the network subsystem immediately decreases to 18 MB, because compression at the kernel socket buffer interface results in a more efficient use of memory. See Section 6.2.3.3 for more information on the sbcompress_threshold attribute.

2.4.5.4 Socket Connections

Each socket results in a network connection. If the system allocates an excessive number of sockets, use the netstat -an command to determine the state of your existing network connections. The following example shows the contents of the protocol control block table and the number of TCP connections currently in each state:

```
# netstat -an | grep tcp | awk '{print $6}' | sort | uniq -c
    1 CLOSE_WAIT
   58 ESTABLISHED
   12 FIN_WAIT_1
    8 FIN_WAIT_2
   17 LISTEN
    1 SYN_RCVD
15749 TIME_WAIT
#
```

For Internet servers, the majority of connections usually are in a TIME_WAIT state. If the number of entries in the FIN_WAIT_1 and FIN_WAIT_2 fields represent a large percentage of the total connections (add together all of the fields), you may want to enable keepalive (see Section 6.3.2.5).

If the number of entries in the SYN_RCVD field represents a large percentage of the total connections, the server may be overloaded or experiencing TCP SYN attacks.

Note that in this example there are almost 16,000 sockets being used, which requires 16 MB of memory. See Section 6.1.2 for more information on configuring memory and swap space.

2.4.5.5 Dropped or Lost Packets

Use the netstat -p ip command to check for bad checksums, length problems, excessive redirects, and packets lost because of resource problems for the IP protocol. Check the output for a nonzero number in the lost packets due to resource problems field. For example:

```
# netstat -p ip
ip:
        259201001 total packets received
        0 bad header checksums
        0 with size smaller than minimum
        0 with data size < data length
        0 with header length < data size
        0 with data length < header length
        25794050 fragments received
```

```
0 fragments dropped (duplicate or out of space)
802 fragments dropped after timeout
0 packets forwarded
67381376 packets not forwardable
        67381376 link-level broadcasts
0 packets denied access
0 redirects sent
0 packets with unknown or unsupported protocol
170988694 packets consumed here
160039654 total packets generated here
0 lost packets due to resource problems
4964271 total packets reassembled ok
2678389 output packets fragmented ok
14229303 output fragments created
0 packets with special flags set
```

Use the `netstat -id` command to monitor dropped output packets. Examine the output for a nonzero value in the `Drop` column. If a nonzero value appears in the `Drop` column for an interface, you may want to increase the value of the `ifqmaxlen` kernel variable to prevent dropped packets. See Section 6.3.2.9 for more information on this attribute.

The following example shows 4,221 dropped output packets on the `tu1` network interface:

```
# netstat -id
Name Mtu  Network Address             Ipkts Ierrs  Opkts Oerrs  Coll  Drop
tu0 1500  Link    00:00:f8:06:0a:b1       0     0  98129 98129     0     0
tu0 1500  DLI     none                    0     0  98129 98129     0     0
tu1 1500  Link    aa:00:04:00:6a:4e  892390   785 814280 68031 93848  4221
tu1 1500  DLI     none               892390   785 814280 68031 93848  4221
tu1 1500  orange  flume              892390   785 814280 68031 93848  4221
  .
  .
  .
```

The output of the previous command shows that the `Opkts` and `Oerrs` fields have the same values for the `tu0` interface, which indicates that the Ethernet cable is not connected. In addition, the value of the `Oerrs` field for the `tu1` interface is 68,031, which is a high error rate. Use the `netstat -is` command to obtain detailed error information.

2.4.5.6 Retransmissions, Out-of-Order Packets, and Bad Checksums

Use the `netstat -p tcp` command to check for retransmissions, out-of-order packets, and bad checksums for the TCP protocol. Use the `netstat -p udp` command to look for bad checksums and full sockets for the UDP protocol. You can use the output of these commands to identify network performance problems by comparing the values in some fields to the total number of packets sent or received.

For example, an acceptable percentage of retransmitted packets or duplicate acknowledgments is 2 percent or less. An acceptable percentage of bad checksums is 1 percent or less.

In addition, a large number of entries in the `embryonic connections dropped` field may indicate that the listen queue is too small or server performance is slow and clients have canceled requests. Other important fields to examine include the `completely duplicate packets`, `out-of-order packets`, and `discarded` fields. For example:

```
# netstat -p tcp
tcp:
        66776579 packets sent
                58018945 data packets (1773864027 bytes)
                54447 data packets (132256902 bytes) retransmitted
                5079202 ack-only packets (3354381 delayed)
                29 URG only packets
                7266 window probe packets
                2322828 window update packets
                1294022 control packets
        40166265 packets received
                29455895 acks (for 1767211650 bytes)
                719524 duplicate acks
                0 acks for unsent data
                19788741 packets (2952573297 bytes) received in-sequence
                123726 completely duplicate packets (9224858 bytes)
                2181 packets with some dup. data (67344 bytes duped)
                472000 out-of-order packets (85613803 bytes)
                1478 packets (926739 bytes) of data after window
                43 window probes
                201331 window update packets
                1373 packets received after close
                118 discarded for bad checksums
                0 discarded for bad header offset fields
                0 discarded because packet too short
        448388 connection requests
        431873 connection accepts
        765040 connections established (including accepts)
        896693 connections closed (including 14570 drops)
        86298 embryonic connections dropped
        25467050 segments updated rtt (of 25608120 attempts)
        106020 retransmit timeouts
                145 connections dropped by rexmit timeout
        6329 persist timeouts
        37653 keepalive timeouts
                15536 keepalive probes sent
                16874 connections dropped by keepalive
```

The output of the previous command shows that, out of the 58,018,945 data packets that were sent, 54,447 packets were retransmitted, which is a percentage that is within the acceptable limit of 2 percent.

In addition, the command output shows that, out of the 29,455,895 acknowledgments, 719,524 were duplicates, which is a percentage that is slightly larger than the acceptable limit of 2 percent.

Important fields for the `netstat -p udp` command include the `incomplete headers`, `bad data length fields`, `bad checksums`, and `full sockets` fields, which should have low values. The `no port` field specifies the number of packets that arrived destined for a nonexistent port (for example, `rwhod` or `routed` broadcast packets) and were subsequently

discarded. A large value for this field is normal and does not indicate a performance problem. For example:

```
# netstat -p udp
udp:
         144965408 packets sent
         217573986 packets received
         0 incomplete headers
         0 bad data length fields
         0 bad checksums
         5359 full sockets
         28001087 for no port (27996512 broadcasts, 0 multicasts)
         0 input packets missed pcb cache
```

The previous example shows a value of 5,359 in the `full sockets` field, which indicates that the UDP socket buffer may be too small.

2.4.5.7 Routing Statistics

Use the `netstat -rs` command to obtain routing statistics. The value of the `bad routing redirects` field should be small. A large value may indicate a serious network problem. For example:

```
# netstat -rs
routing:
         0 bad routing redirects
         0 dynamically created routes
         0 new gateways due to redirects
         1082 destinations found unreachable
         0 uses of a wildcard route
```

2.4.5.8 Protocol Statistics

Use the `netstat -s` command to simultaneously display statistics related to the IP, ICMP, IGMP, TCP, and UDP protocol layers. For example:

```
# netstat -s
ip:
         377583120 total packets received
         0 bad header checksums
         7 with size smaller than minimum
         0 with data size < data length
         0 with header length < data size
         0 with data length < header length
         12975385 fragments received
         0 fragments dropped (dup or out of space)
         3997 fragments dropped after timeout
         523667 packets forwarded
         108432573 packets not forwardable
         0 packets denied access
         0 redirects sent
         0 packets with unknown or unsupported protocol
         259208056 packets consumed here
         213176626 total packets generated here
         581 lost packets due to resource problems
         3556589 total packets reassembled ok
         4484231 output packets fragmented ok
         18923658 output fragments created
         0 packets with special flags set
icmp:
         4575 calls to icmp_error
```

```
        0 errors not generated because old ip message was too short
        0 errors not generated because old message was icmp
        Output histogram:
                echo reply: 586585
                destination unreachable: 4575
                time stamp reply: 1
        0 messages with bad code fields
        0 messages < minimum length
        0 bad checksums
        0 messages with bad length
        Input histogram:
                echo reply: 612979
                destination unreachable: 147286
                source quench: 10
                echo: 586585
                router advertisement: 91
                time exceeded: 231
                time stamp: 1
                time stamp reply: 1
                address mask request: 7
        586586 message responses generated
igmp:
        0 messages received
        0 messages received with too few bytes
        0 messages received with bad checksum
        0 membership queries received
        0 membership queries received with invalid field(s)
        0 membership reports received
        0 membership reports received with invalid field(s)
        0 membership reports received for groups to which we belong
        0 membership reports sent
tcp:
        66818923 packets sent
                58058082 data packets (1804507309 bytes)
                54448 data packets (132259102 bytes) retransmitted
                5081656 ack-only packets (3356297 delayed)
                29 URG only packets
                7271 window probe packets
                2323163 window update packets
                1294434 control packets
        40195436 packets received
                29477231 acks (for 1797854515 bytes)
                719829 duplicate acks
                0 acks for unsent data
                19803825 packets (2954660057 bytes) received in-sequence
                123763 completely duplicate packets (9225546 bytes)
                2181 packets with some dup. data (67344 bytes duped)
                472188 out-of-order packets (85660891 bytes)
                1479 packets (926739 bytes) of data after window
                43 window probes
                201512 window update packets
                1377 packets received after close
                118 discarded for bad checksums
                0 discarded for bad header offset fields
                0 discarded because packet too short
        448558 connection requests
        431981 connection accepts
        765275 connections established (including accepts)
        896982 connections closed (including 14571 drops)
        86330 embryonic connections dropped
        25482179 segments updated rtt (of 25623298 attempts)
        106040 retransmit timeouts
                145 connections dropped by rexmit timeout
        6329 persist timeouts
```

```
        37659 keepalive timeouts
                15537 keepalive probes sent
                16876 connections dropped by keepalive
udp:
        145045792 packets sent
        217665429 packets received
        0 incomplete headers
        0 bad data length fields
        0 bad checksums
        5359 full sockets
        28004209 for no port (27999634 broadcasts, 0 multicasts)
        0 input packets missed pcb cache
```

2.4.6 Gathering NFS Server Side Information Using ps axlmp

On an NFS server system, the `nfsd` daemon spawns several I/O threads to service I/O requests from clients. A sufficient number of threads must be configured to handle the number of concurrent requests typical for the server. The default configuration of eight UDP and eight TCP threads is enough for a workstation exporting a small number of directories to a handful of clients. For a heavily used NFS server, up to 128 server threads can be configured and distributed over TCP and UDP. Monitor the NFS server threads on your server to determine if more threads are required to service the NFS load.

To display idle I/O threads on a server system, enter:

```
# /usr/ucb/ps axlmp 0 | grep -v grep | grep -c nfs_udp

# /usr/ucb/ps axlmp 0 | grep -v grep | grep -c nfs_tcp
```

This will display a count of the number of sleeping and idle UDP and TCP threads, respectively. If the number of sleeping and idle threads is zero or lower, you might improve NFS performance by increasing the number of threads. See Section 5.4.1 or `nfsd`(8) for more information.

2.4.7 Gathering NFS Client Side Information Using nfsiod

On an NFS client system, the `nfsiod` daemon spawns several I/O threads to service asynchronous I/O requests to the server. The I/O threads improve the performance of both NFS read and writes. The optimum number of I/O threads depends on many variables, such as how quickly the client will be writing, how many files will be accessed simultaneously, and the characteristics of the NFS server. For small clients, the default of seven threads is sufficient. For larger servers with heavy NFS client activity, more client threads may be necessary. Monitor the NFS client threads on your server to determine if more threads are required to service the NFS load.

To display idle I/O threads on a client system, enter:

```
# ps axlm | grep -v grep | grep -c nfsiod
```

This will display a count of the number of sleeping and idle I/O threads. If the number of sleeping and idle threads is often zero or lower, you might

improve NFS performance by increasing the number of threads. See Section 5.4.2 or nfsiod(8) for more information.

2.4.8 Monitoring Incoming Network Traffic to an NFS Server by Using the nfswatch Command

The nfswatch program monitors all incoming network traffic to a NFS file server and divides it into several categories. The number and percentage of packets received in each category is displayed on the screen in a continuously updated display. The nfswatch command can usually be run without options and will produce useful results. For example:

```
#  /usr/sbin/nfswatch
Interval packets:            628 (network)          626 (to host)         0 (dropped)
Total packets:              6309 (network)         6307 (to host)         0 (dropped)
                              Monitoring packets from interface ee0
                      int   pct    total                          unt   pct    total
ND Read                0     0%       0     TCP Packets            0    0%        0
ND Write               0     0%       0     UDP Packets          139   10%      443
NFS Read               2     0%       2     ICMP Packets           1    0%        1
NFS Write              0     0%       0     Routing Control       81    6%      204
NFS Mount              2     0%       3     Address Resolution   109    8%      280
YP/NIS/NIS+            0     0%       0     Reverse Addr Resol    15    1%       43
RPC Authorization      7     1%      12     Ethernet/FDDI Bdcst  406   30%     1152
Other RPC Packets      4     0%      10     Other Packets       1087   80%     3352

           18 NFS Procedures [10 not displayed]              more->
 Procedure            int   pct   total  completed  avg(msec)  std dev   max resp
  CREATE                0    0%      0
  GETATTR               1   50%      1         1      3.90                3.90
  GETROOT               0    0%      0
  LINK                  0    0%      0
  LOOKUP                0    0%      0
  MKDIR                 0    0%      0
  NULLPROC              0    0%      0
  READ                  0    0%      0
```

Note

The nfswatch command monitors and displays data for only file systems mounted with NFS Version 2.0.

Your kernel must be configured with the packetfilter option. After kernel configuration, any user can invoke nfswatch once the superuser has enabled promiscuous-mode operation using the pfconfig command. See packetfilter(7) for more information.

2.5 Additional Tools for Monitoring Performance

You may want to set up a routine to continuously monitor system performance. Some monitoring tools will alert you when serious problems

occur (for example, mail). It is important that you choose a monitoring tool that has low overhead to obtain accurate performance information.

Table 2–1 describes the tools that you use to continuously monitor performance.

Table 2–1: Tools for Continuous Performance Monitoring

Name	Description
Performance Visualizer	Graphically displays the performance of all significant components of a parallel system. Using Performance Visualizer, you can monitor the performance of all the member systems in a cluster. It monitors the performance of several systems simultaneously, it allows you to see the impact of a parallel application on all the systems, and to ensure that the application is balanced across all systems. When problems are identified, you can change the application code and use Performance Visualizer to evaluate the impact of these changes. Performance Visualizer is a Tru64 UNIX layered product and requires a license. It also helps you identify overloaded systems, underutilized resources, active users, and busy processes. You can choose to look at all of the hosts in a parallel system or at individual hosts. See the Performance Visualizer documentation for more information.
`monitor`	Collects a variety of performance data on a running system and either displays the information or saves it to a binary file. The `monitor` utility is available on the Tru64 UNIX Freeware CD-ROM. See **`http://www.tru64unix.compaq.com/demos/ossc-v51a/html/monitor.htm`** for more information.
`top`	Provides continuous reports on the state of the system, including a list of the processes using the most CPU resources. The `top` command is available on the Tru64 UNIX Freeware CD-ROM. See **`ftp://eecs.nwu.edu/pub/top`** for more information.
`xload`	Displays the system load average in a histogram that is periodically updated. See `xload`(1X) for information.

Table 2–1: Tools for Continuous Performance Monitoring (cont.)

Name	Description
`volstat`	Provides information about activity on volumes, plexes, subdisks, and disks under LSM control. The `volstat` utility reports statistics that reflect the activity levels of LSM objects since boot time or since you reset the statistics. See Section 9.3 for information.
`volwatch`	Monitors LSM for failures in disks, volumes, and plexes, and sends mail if a failure occurs. See Section 9.3 for information.

2.6 Gathering Profiling and Debugging Information

You can use profiling to identify sections of application code that consume large portions of execution time, and you can use these tools to profile and debug the kernel. To improve performance, concentrate on improving the coding efficiency of those time-intensive sections.

Table 2–2 describes the commands you can use to obtain information about applications. Detailed information about these tools is located in the *Programmer's Guide* and the *Kernel Debugging* manual.

In addition, `prof_intro`(1) provides an overview of application profilers, profiling, optimization, and performance analysis.

Table 2–2: Application Profiling and Debugging Tools

Name	Use	Description
`atom`	Profiles applications	Consists of a set of prepackaged tools (`third`, `hiprof`, or `pixie`) that can be used to instrument applications for profiling or debugging purposes. The `atom` toolkit also consists of a command interface and a collection of instrumentation routines that you can use to create custom tools for instrumenting applications. See the *Programmer's Guide* and `atom`(1) for more information.

Table 2–2: Application Profiling and Debugging Tools (cont.)

Name	Use	Description
`third`	Checks memory access and detects memory leaks in applications	Performs memory access checks and memory leak detection of C and C++ programs at run time, by using the `atom` tool to add code to executable and shared objects. The Third Degree tool instruments the entire program, including its referenced libraries. See `third`(1) for more information.
`hiprof`	Produces a profile of procedure execution times in an application	An `atom`-based program profiling tool that produces a flat profile, which shows the execution time spent in any given procedure, and a hierarchical profile, which shows the time spent in a given procedure and all of its descendents. The `hiprof` tool uses code instrumentation instead of program counter (PC) sampling to gather statistics. The `gprof` command is usually used to filter and merge output files and to format profile reports. See `hiprof`(1) for more information.
`pixie`	Profiles basic blocks in an application	Produces a profile showing the number of times each instruction was executed in a program. The information can be reported as tables or can be used to automatically direct later optimizations by using the `-feedback, -om,` or `-cord` options in the C compiler (see `cc`(1)). The `pixie` profiler reads an executable program, partitions it into basic blocks, and writes an equivalent program containing additional code that counts the execution of each basic block. The `pixie` utility also generates a file containing the address of each of the basic blocks. When you run this `pixie`-generated program, it generates a file containing the basic block counts. The `prof` and `pixstats` commands can analyze these files. See `pixie`(1) for more information.

Table 2–2: Application Profiling and Debugging Tools (cont.)

Name	Use	Description
`prof`	Analyzes profiling data	Analyzes profiling data and produces statistics showing which portions of code consume the most time and where the time is spent (for example, at the routine level, the basic block level, or the instruction level). The `prof` command uses as input one or more data files generated by the `kprofile`, `uprofile`, or `pixie` profiling tools. The `prof` command also accepts profiling data files generated by programs linked with the `-p` switch of compilers such as `cc`. The information produced by `prof` allows you to determine where to concentrate your efforts to optimize source code. See `prof`(1) for more information.
`gprof`	Analyzes profiling data and displays procedure call information and statistical program counter sampling in an application	Analyzes profiling data and allows you to determine which routines are called most frequently, and the source of the routine call, by gathering procedure call information and performing statistical program counter (PC) sampling. The `gprof` tool produces a flat profile of the routines' CPU usage. To produce a graphical execution profile of a program, the tool uses data from PC sampling profiles, which are produced by programs compiled with the `cc -pg` command, or from instrumented profiles, which are produced by programs modified by the `atom -tool hiprof` command. See `gprof`(1) for more information.
`uprofile`	Profiles user code in an application	Profiles user code using performance counters in the Alpha chip. The `uprofile` tool allows you to profile only the executable part of a program. The `uprofile` tool does not collect information on shared libraries. You process the performance data collected by the tool with the `prof` command. See the *Kernel Debugging* manual or `uprofile`(1) for more information.

Table 2–2: Application Profiling and Debugging Tools (cont.)

Name	Use	Description
`kprofile`	Produces a program counter profile of a running kernel	Profiles a running kernel using the performance counters on the Alpha chip. You analyze the performance data collected by the tool with the `prof` command. See `kprofile`(1) for more information.
Visual Threads	Identifies bottlenecks and performance problems in multithreaded applications	Enables you to analyze and refine your multithreaded applications. You can use Visual Threads to identify bottlenecks and performance problems, and to debug potential thread-related logic problems. Visual Threads uses rule-based analysis and statistics capabilities and visualization techniques. Visual Threads is licensed as part of the Developers' Toolkit for Tru64 UNIX.
`dbx`	Debugs running kernels, programs, and crash dumps, and examines and temporarily modifies kernel variables	Provides source-level debugging for C, Fortran, Pascal, assembly language, and machine code. The `dbx` debugger allows you to analyze crash dumps, trace problems in a program object at the source-code level or at the machine-code level, control program execution, trace program logic and flow of control, and monitor memory locations. Use `dbx` to debug kernels, debug stripped images, examine memory contents, debug multiple threads, analyze user code and applications, display the value and format of kernel data structures, and temporarily modify the values of some kernel variables. See `dbx`(8) for more information.
`kdbx`	Debugs running kernels and crash dumps	Allows you to examine a running kernel or a crash dump. The `kdbx` debugger, a frontend to the `dbx` debugger, is used specifically to debug kernel code and display kernel data in a readable format. The debugger is extensible and customizable, allowing you to create commands that are tailored to your kernel debugging needs. You can also use extensions to check resource usage (for example, CPU usage). See `kdbx`(8) for more information.

Table 2–2: Application Profiling and Debugging Tools (cont.)

Name	Use	Description
`ladebug`	Debugs kernels and applications	Debugs programs and the kernel and helps locate run-time programming errors. The `ladebug` symbolic debugger is an alternative to the `dbx` debugger and provides both command-line and graphical user interfaces and support for debugging multithreaded programs. See the *Ladebug Debugger Manual* and `ladebug`(1) for more information.
`lsof`	Displays open files	Displays information about files that are currently opened by the running processes. The `lsof` is available on the Tru64 UNIX Freeware CD-ROM.

3

Displaying and Modifying Kernel Subsystem Attributes

The operating system includes various subsystems that define or extend the kernel. Kernel subsystem attributes are used to set kernel variables, which control subsystem behavior or track subsystem statistics. Attributes are assigned default values at boot time. If the default values of some attributes are not appropriate for your system, you must modify these values to provide optimal performance.

To display and modify kernel subsystem attributes, follow these steps:

1. Determine the operating system support for an attribute (Section 3.1)
2. Display the attribute values (Section 3.2)
3. Modify the attribute values (Section 3.3)

3.1 Operating System Support for Attributes

To determine if your version of the operating system supports a particular kernel subsystem attribute, use the following method:

- Use the `sysconfig -q` *subsystem* [*attribute*] command.

  If you do not specify an attribute, the system displays all the subsystem attributes that can be modified with the `sysconfig` or `sysconfigdb` command. If the subsystem is not configured, the operating system displays a message similar to the following:

  ```
  framework error: subsystem 'inet' not found
  ```

  If you specify an attribute, only the information specific to that attribute is displayed. For example:

  ```
  # sysconfig -q inet tcbhashsize
  inet:
  tcbhashsize = 32
  ```

  If the attribute is not supported or if it cannot be accessed by using `sysconfig`, the operating system displays a message similar to the following:

  ```
  inet:
  tcbhashsize = unknown attribute
  ```

- Use the `dbx p` (print) command. If you cannot access the attribute, the operating system displays a message stating that the attribute is not defined or active. For example:

```
# dbx -k /vmunix
dbx version 5.1
Type 'help' for help.

(dbx) p tcp_keepalive_default
"tcp_keepalive_default" is not defined or not active
(dbx)
```

See `sysconfig`(8) and `dbx`(1) for more information.

3.2 Displaying Attribute Values

There are various methods that you can use to display the current value of a kernel subsystem attribute and other descriptive information. Use:

- The Kernel Tuner (`dxkerneltuner`) graphical user interface (GUI) to display permanent, current (run-time), minimum, and maximum values of attributes. Access the GUI through the Common Desktop Environment (CDE) Application Manager window; select the `System_Admin` icon, and then select the `MonitoringTuning` icon. You can then choose the subsystem whose attributes you want to display.
- The `sysconfig -q` *subsystem* [*attribute*] command to display the current (run-time) value of the specified attribute or, if an attribute is not specified, all the attributes for the specified subsystem:

  ```
  sysconfig -q subsystem [attribute]
  ```

  For example:

  ```
  # sysconfig -q vm ubc_maxpercent
  vm:
  ubc_maxpercent = 100
  ```

- The `sysconfig -Q` *subsystem* [*attribute*] command to display the maximum and minimum values of the attributes for the specified subsystem. If you specify a particular attribute, the system displays information only for that attribute.

  The command output also includes information about the operations that you can perform on the attribute. The following list describes some of the `sysconfig` attributes:

  – `C` - The attribute can be modified when the subsystem is initially loaded; that is, the attribute supports boot time, permanent modifications.

  – `R` - The attribute can be tuned at run time; that is, you can modify the value that the system is currently using.

  – `Q` - The attribute's current value can be displayed (queried).

For example:

```
# sysconfig -Q vfs bufcache
vfs:
bufcache -     type=INT op=CQ min_val=0 max_val=50
```

The output of the previous command shows that the minimum value of the `bufcache` attribute is 0 and the maximum value is 50. The output also shows that you cannot modify the current (run-time) value.

The following example shows how to use the `dbx p` (print) command to display the current value of a kernel variable, instead of an attribute. For example:

```
# dbx -k /vmunix
dbx version 5.1
Type 'help' for help.

(dbx) p ipport_userreserved
5000
(dbx)
```

See `dxkerneltuner`(8), `sysconfig`(8), and `dbx`(1) for more information.

3.3 Modifying Attribute Values

There are various methods that you can use to modify attribute values. The method you use depends on the version of the operating system you are running and whether you want to modify the current (run-time) value of an attribute or modify an attribute's permanent value. You must be root to modify attribute values.

The following sections describe how to modify the current and permanent values.

3.3.1 Current Value

In some cases, you can modify the current (run-time) value of an attribute. This allows you to determine if modifying an attribute will improve your system performance without rebooting the system. Not all attributes can be tuned at run time, and the temporary modifications are lost when you reboot the system. Use the `sysconfig -Q` command to determine whether an attribute can be tuned at run time.

To modify an attribute's current (run-time) value, use one of the following methods:

- The Kernel Tuner (`dxkerneltuner`) GUI, if the attribute supports this operation. Access the GUI through the Common Desktop Environment (CDE) Application Manager window; select the `System_Admin` icon, and then select the `MonitoringTuning` icon. Choose the subsystem whose

attribute you want to modify, and enter the new value in the Current Value field.

- The sysconfig -r command, if the attribute supports this operation. Use the following command syntax:

  ```
  sysconfig -r subsystem attribute=value
  ```

  For example:

  ```
  # sysconfig -r inet tcp_keepinit=30
  tcp_keepinit: reconfigured
  ```

The following example shows how to use the dbx assign command to modify the current value of a kernel variable, instead of an attribute. However, modifications made with the dbx assign command are lost when you reboot the system. Use the following command syntax:

```
dbx assign attribute=value
```

For example:

```
# dbx -k /vmunix
dbx version 5.1
Type 'help' for help.

(dbx) assign ipport_userreserved=60000
60000
(dbx)
```

See dxkerneltuner(8), sysconfig(8), and dbx(1) for more information.

3.3.2 Permanent Value

To modify an attribute's permanent (boot-time) value, the sysconfigtab file must contain the subsystem name, the attribute name, and the value of the attribute. Do not manually modify the sysconfigtab file. To make these modifications, use one of the following methods:

- The Kernel Tuner (dxkerneltuner) GUI. Access the GUI through the Common Desktop Environment (CDE) Application Manager window, select the System_Admin icon, and then select the MonitoringTuning icon. Choose the subsystem whose attribute you want to modify, and enter the new value in the Boot Time Value field.
- The sysconfigdb command. Use the following command syntax:

  ```
  sysconfigdb -a -f stanza_file subsystem
  ```

  The *stanza_file* is a specially formatted file that contains the name of the subsystem and a list of attributes and their values. This file is merged into the sysconfigtab file. See stanza(4) for more information.

  To use the new attribute value, you must invoke the sysconfig -r command if the attribute can be tuned at run time, or reboot the system.

In addition, you can use the `dbx patch` command to modify the value of a variable, as well as the on-disk `/vmunix` image value.

See `dxkerneltuner`(8), `sysconfig`(8), and `sysconfigdb`(8) for more information. See the *System Administration* manual for information about modifying the system configuration file.

Part 2

Tuning by Application

4

Tuning Oracle

This chapter describes how to improve your Oracle 8.1.7.x/9*i* database performance. It describes several monitoring tools and offers tuning recommendations, including:

- Monitoring Oracle statistics (Section 4.1)
- Improving the performance of the `gettimeofday( )` function (Section 4.2)
- Choosing and enabling IPC communication protocols (Section 4.3)

Note

This manual assumes that you are using Oracle Version 8.1.7 or higher. This version requirement is important because Oracle Version 8.1.7 and higher, use the direct I/O capabilities of AdvFS to bypass the Unified Buffer Cache (UBC) part of the file system layers.

4.1 Monitoring Oracle Statistics

There are several commands and utilities that you can use to gather system performance information. It is important that you gather statistics under a variety of conditions. Comparing sets of data will help you to diagnose performance problems.

Table 4–1 describes the tools that you can use to monitor a system running an Oracle application.

Table 4–1: Tools to Detect Poor Oracle Application Performance

Tools	Description	Reference
`collect`	Records or displays specific operating system data. It also gathers the vital system performance information for specific subsystems.	Section 2.3.2

Table 4–1: Tools to Detect Poor Oracle Application Performance (cont.)

Tools	Description	Reference
`lockinfo`	Collects and displays locking statistics for the kernel SMP locks. It uses the `/dev/lockdev` pseudo driver to collect data.	Section 2.4.1
`sched_stat`	Helps determine how well the system load is distributed among CPUs, what kinds of jobs are getting or not getting sufficient cycles on each CPU, and how well cache affinity is being maintained for these jobs.	Section 2.4.2

See `collect`(8), `lockinfo`(8), and `sched_stat`(8) for more information.

4.2 Improving the Performance of the gettimeofday( ) Function

The Oracle server times many functions as it executes. This is especially true if the `INIT.ORA` parameter `timed_statistics` is set to `TRUE`.

These timing functions result in system calls into the operating system kernel, which can degrade Oracle performance because the calling process relinquishes the CPU. There is a feature in Tru64 UNIX that gives a process direct access to the operating system's real-time clock.

Using this feature will improve performance on a heavily used system. It will also improve performance on a lightly loaded system, but it may not be as noticeable.

Note

This feature is supported on Oracle Version Version 7.3 and higher.

To enable this feature, enter the following commands:

```
# mknod /dev/timedev c 150

# chmod 644 /dev/timedev
```

If you are working in a cluster, enter the commands on each cluster member. The `/dev/timedev` special file will be persistent across system reboots.

To use this feature with Oracle, the instance has to be restarted. The existence of the `/dev/timedev` file is checked only on instance startup. We recommend that all instances in a cluster (and hence all nodes) have this feature enabled.

See `gettimeofday`(2) for more information.

4.3 Choosing and Enabling IPC Communication Protocols

Oracle can use either UDP or RDG (Reliable Datagram) for DLM/IPQ interinstance communication.We recommend using RDG as the protocol for IPC instead of using UDP.

Note

Although Oracle 8.1.7 does support the use of RDG for communication, it is not recommended to enable it but continue to use UDP. In some cases it might be necessary to use UDP as the communication protocol in a Oracle 9*i* environment.

The following commands show how to enable and disable the different protocols for IPC.

Use the following command to disable NUMA support on Oracle 8*i* or 9*i*:

```
# cd $ORACLE_HOME/rdbms/lib
# make -f ins_rdbms.mk numa_off
# make -f ins_rdbms.mkioracle
```

Enabling NUMA support in Oracle is currently not supported for Oracle 8.1.7/OPS and 9.0.1/RAC installations. If you are planning to use RAC or OPS you must disable NUMA.

Use the following commands to enable Oracle 9*i* RAC or 8.1.7 Oracle Parallel Server:

- In all cases:

  ```
  # cd $ORACLE_HOME/rdbms/lib
  ```

- For Oracle 8*i*:

  ```
  # make -f ins_rdbms.mk ops_on
  # make -f ins_rdbms.mk ioracle
  ```

- For Oracle 9*i*:

  ```
  # make -f ins_rdbms.mk rac_on
  # make -f ins_rdbms.mk ioracle
  ```

- To make IPC use the UDP Protocol (default for Oracle 8*i* OPS):

  ```
  # make -f ins_rdbms.mk ipc_udp
  # make -f ins_rdbms.mk ioracle
  ```

- To use reliable datagram (RDG) for IPC (default for Oracle 9*i*):

```
# make -f ins_rdbms.mk rac_on
# make -f ins_rdbms.mk ioracle
```

4.4 Tuning Recommendations

There are many kernel subsystem attributes that affect Oracle 8.1.7.x/9*i* database performance. This section offers primary tuning recommendations for some of the attributes for the following subsystems:

- Virtual Memory (Section 4.4.1)
- Advanced File System (Section 4.4.2)
- Virtual File System (Section 4.4.3)
- Interprocess Communication (Section 4.4.4)
- Internet (Section 4.4.5)
- Process (Section 4.4.6)
- Real time (Section 4.4.7)
- Reliable Datagram (Section 4.4.8)
- Memory Channel (Section 4.4.9)

Note

Some kernel subsystem attributes enable you to modify their value and apply the value to a running system. Other attributes require you to reboot the system to use a new value. See Section 3.3.1 to determine if an attribute can be tuned at run time.

See `sys_attrs`(5) for more information.

4.4.1 Modifying Virtual Memory Attributes

You may be able to improve Oracle 8.1.7.x/9*i* database performance by tuning the following `vm` subsystem attributes:

- `new_wire_method` (Section 4.4.1.1)
- `rad_gh_regions` (Section 4.4.1.2)
- `gh_chunks` (Section 4.4.1.2.2)
- `ubc_maxpercent` (Section 4.4.1.3)
- `ubc_borrowpercent` (Section 4.4.1.4)
- `vm_ubcseqstartpercent` (Section 4.4.1.6)
- `vm_ubcdirtypercent` (Section 4.4.1.7)

- vm_swap_eager (Section 4.4.1.8)

See sys_attrs_vm(5) for more information and see Chapter 3 for information about modifying kernel subsystem attributes.

4.4.1.1 Disabling Shared Memory

Using granularity hints shared memory, often referred to as large pages, is the recommended method of setting shared memory for Oracle 8.1.7.x/9*i*. However, using segmented shared memory (SSM), which is controlled by the ssm_threshold attribute in the ipc subsystem, is also supported and is enabled by default.

Set new_wire_method to 0 if SSM is used. For example, if the ssm_threshold has a default value of 8 MB, set new_wire_method to 0.

There is an interaction between the asynchronous I/O (AIO) and page wiring mechanism that can result in high system time. To solve this issue, we recommend setting the new_wire_method attribute to 0. The new_wire_method default is 1.

Disabling this tunable attribute does not result in any negative performance impacts. However, note that sys_attrs_vm(5) tells you to not modify the default setting (1, on) for this attribute unless instructed to do so by support personnel or the patch kit documentation.

One way to avoid the interaction between new_wire_method and ssm_threshold is to use granularity hints memory (gh_regions or rad_gh_regions) to prewire memory for the database to use.

See Section 4.4.1.2.2 and Section 4.4.1.2.1 for more information on gh_chunks and rad_gh_regions.

4.4.1.2 Allocating Shared Memory

Two options for allocating shared memory for Oracle 8.1.7.x/9*i* are supported. The traditional option is to use segmented shared memory (SSM), which is controlled through the tunable ssm_threshold attribute in the ipc subsystem. The ssm_threshold attribute is enabled by default.

Using granularity hints such as gh_chunks, often referred to as large pages, is the preferred method of allocating shared memory on GS80/160/320 systems running Oracle 8.1.7.x/9*i*. (Granularity hints attributes are documented in sys_attrs_vm(5).) This memory cannot be used for any other purpose, and it cannot be returned to the system or reclaimed when not being used. It will be used solely by the database for its system global area (SGA), or by any other application using shmget( ) or nshmget( ) to allocate shared memory.

Use the `vmstat -P` command to determine the amount of granularity hints memory configured on a system. The two attributes that determine how chunks of memory are reserved for shared memory are `rad_gh_regions` and `gh_chunks`. The use of `rad_gh_regions` and `gh_chunks`, change depending on which system you are running:

- NUMA-aware systems (GS80, GS160, GS320) should set the `rad_gh_regions` attribute. Although the `gh_chunks` attribute applies to both non-NUMA and NUMA-aware systems, on NUMA-aware systems the `gh_chunks` attribute affects only the first Resource Affinity Domain (RAD) supported by the system (see Section 4.4.1.2.1).
- We recommend setting the `gh_chunks` attribute if you want to use granularity hints memory allocation on non-NUMA systems such as the ES, DS, and GS140 platforms (see Section 4.4.1.2.2).

4.4.1.2.1 Modifying the rad_gh_regions Attribute

To set the `rad_gh_regions` attribute on GS80/GS160/GS320 platforms, specify the amount of memory per RAD/QBB, in MB, through the corresponding `rad_gh_regions` attribute. For example, to allocate 2 GB on QBB0 change the `rad_gh_regions` attribute to 2048 MB.

To determine the `rad_gh_regions` setting, take your planned or projected Oracle SGA size and divide by the number of QBBs/RAD's your GS system has configured. For each `rad_gh_regions[x]` (where `x` represents the QBB identifier from 0-7), specify the required value in MB. For example, if you are running a GS320 platform that has 64 GB of main memory in 8 QBBs, and your Oracle SGA is sized at 16 GB, change the value to at least 2048 MB for `rad_gh_regions[0]` to `rad_gh_regions[7]`.

We recommend that the sum of `rad_gh_regions[*]` be set to at least the size of the Oracle SGA. However, you might consider allocating a larger value for `rad_gh_regions` in order to resize the Oracle SGA without having to reboot the system. Changes to `rad_gh_regions` require a system reboot and `rad_gh_regions` is not a dynamic system tunable.

Shared memory should generally be allocated in striped mode in order to distribute memory across all available RADs. Changing to a sequential allocation policy may adversely affect performance as it may cause hotspots in individual RADs.

If you set `rad_gh_regions` to any number except 0, you must also disable `ssm` by setting `ssm_threshold` to 0. The default value of the `rad_gh_regions` attribute is 0 or disabled.

See Section 4.4.1.2.2 for more information about granularity hints memory allocation.

4.4.1.2.2 Modifying the gh_chunks Attribute

The gh_chunks attribute specifies the number of 4-MB chunks of memory reserved at boot time for shared memory use. This memory cannot be used for any other purpose, nor can it be returned to the system or reclaimed when not being used.

There is only about a 7 percent overall performance improvement when using gh_chunks. Therefore, it might not be the best option for most systems due to its complexity to implement.

However, some systems might still benefit from using gh_chunks, especially Oracle 8.1.7.x/9*i* environments that are having a very large number of clients connecting and disconnecting from the database. In this type of environment, using gh_chunks can actually result in a significant performance increase. We recommend setting gh_chunks to at least the Oracle SGA size and dividing it by 4 MB. Calculate the value for gh_chunks by dividing Oracle's SGA size by 4 MB and then expressing this value in 4-MB units. For example, if your Oracle SGA is sized at 16GB, divide 16 GB by 4 MB, for a result of 4000 MB. This equals setting gh_chunks to 1000. If you set gh_chunks to any number except 0, you must also disable ssm by setting ssm_threshold to 0. The default value of gh_chunks is 0 or disabled.

4.4.1.3 Modifying the Percentage of Physical Memory the UBC is Using

The ubc_maxpercent attribute specifies the maximum percentage of physical memory that the UBC can use at one time.

Oracle 8.1.7.x/9*i* uses the AdvFS direct I/O; therefore you do not need to artificially restrict the Unified Buffer Cache (UBC). We recommend decreasing the physical memory that the UBC can use at one time to a smaller value to prevent (double) caching in the file system. However, setting the ubc_maxpercent to a low value may cause contention in the kernel and negatively impact performance.

We recommend increasing the percentage of physical memory that the UBC can use at one time to at least 70 percent. The value should not be set to smaller than 35 percent.

4.4.1.4 Modifying the Percentage of Memory the UBC is Borrowing

The ubc_borrowpercent attribute specifies the percentage of memory above which the UBC is only borrowing memory from the virtual memory subsystem. Paging does not occur until the UBC has returned all its borrowed pages.

The ubc_borrowpercent default value is 20 percent, which is a good percentage for most systems. However, if you are running a database server

without any current interactive users, consider decreasing the value to 10 percent to improve backup performance.

4.4.1.5 Modifying the Percentage of Memory the UBC Can Use For a Single File

The vm_ubcseqpercent attribute specifies the maximum percentage of UBC memory that can be used to cache a single file. See Section 4.4.1.6 for more information about controlling when the UBC checks this limit. The vm_ubcseqpercent default value is 10 percent, which is a good percentage for most systems. However, if the system you are running is a database server-only environment, consider decreasing the value to 5 percent to improve performance during backups.

4.4.1.6 Modifying the UBC Threshold

The vm_ubcseqstartpercent attribute specifies a threshold value (a percentage of the UBC in terms of its current size) that determines when the UBC starts to check the percentage of UBC pages cached for each file object. If the cached page percentage for any file exceeds the value of vm_ubcseqpercent, the UBC returns that file's UBC LRU pages to virtual memory. See Section 4.4.1.5 for more information about the vm_ubcseqpercent attribute.

Note

The vm_ubcseqstartpercent attribute is defined as a percentage of the ubc_maxpercent attribute, which is itself a percentage of available memory. The definition change has no effect if the ubc_maxpercent specifies its default value (100 percent). However, the definition change has implications if the value for the ubc_maxpercent has been lowered. For example, the value for vm_ubcseqstartpercent should be set to 25.

4.4.1.7 Modifying the Percentage of Pages that Must be Dirty

The vm_ubcdirtypercent attribute specifies the percentage of pages that must be dirty (modified) before the UBC starts writing them to disk. For most systems the default value of 10 percent is effective. However, if you are running a system with a lot of file system/UBC activity and that would benefit from keeping file system pages in the UBC, increase the value to 90 percent.

4.4.1.8 Modifying the Swap Allocation Mode

The vm_swap_eager attribute controls how the system will use the available swap space by specifying the swap allocation mode, which can be

immediate mode (1) or deferred mode (0). There is no performance benefit attached to either mode.

The swap space allocation modes are as follows:

- **Immediate mode** — This mode reserves swap space when a process first allocates anonymous memory. Immediate mode is the default swap space allocation mode and is also called **eager mode**.

  Immediate mode may cause the system to reserve an unnecessarily large amount of swap space for processes. However, it ensures that swap space will be available to processes if it is needed. Immediate mode is recommended for systems that overcommit memory (that is, systems that page).

- **Deferred mode** — This mode reserves swap space only if the virtual memory subsystem needs to write a modified virtual page to swap space. It postpones the reservation of swap space for anonymous memory until it is actually needed. Deferred mode is also called **lazy mode**.

  Deferred mode requires less swap space than immediate mode and may cause the system to run faster because it requires less swap space bookkeeping. However, because deferred mode does not reserve swap space in advance, the swap space may not be available when a process needs it, and the process may be killed asynchronously. Deferred mode is recommended for large-memory systems or systems that do not overcommit memory (page).

If you set the `vm_swap_eager` attribute to 1, the default, your system is in eager swap allocation mode. If you are in this mode, sum the anonymous virtual memory for all processes and add at least 10 percent to the size of your swap space. Use eager swap allocation mode for highly reliable systems that overcommit memory.

If you set the `vm_swap_eager` attribute to 0, your system is in lazy swap allocation mode. If you are in this mode, sum the anonymous virtual memory for all processes and subtract half of the physical memory. Use lazy mode for any system that does not overcommit memory. If `vm_swap_eager` has been set to 0 and the system is in danger of running out of available swap space, a process that attempts to allocate swap space is killed. There is no mechanism to protect processes being deleted in this situation.

We recommend that the Oracle 8.1.7.x/9*i* database server environment have enough memory available to set `vm_swap_eager` to 1. However, if the workload is well understood and the system has been configured with enough memory to prevent it from swapping, you can set `vm_swap_eager` to 0. For eager swap allocation mode, the default value is 1.

4.4.2 Modifying the Advanced File System Attribute

You may be able to improve Oracle 8.1.7.x/9*i* database performance by tuning the `advfs` subsystem `AdvfsSyncMmapPages` attribute.

The `AdvfsSyncMmapPages` attribute specifies a value that controls whether modified (dirty) memory-mapped pages are flushed to disk during a `sync( )` system call. If the value is 1, the dirty memory-mapped pages are asynchronously written to disk. If the value is 1, dirty memory-mapped pages are not written to disk during a sync system call.

Setting the parameter to 0 prevents AdvFS from trying to flush pages of files that have been mmapped. the default value of one causes memory-mapped pages to be written asynchronously to disk during a `sync( )` system call.

Most applications that use `mmap( )` to map pages and files into memory are using their own synchronization through the `fsync( )` call, so there is no need for AdvFS to perform the same operation again. This setting also avoids AdvFS trying to flush pages that should actually stay in memory.

See `sys_attrs_advfs`(5) for more information and see Chapter 3 for information about modifying kernel subsystem attributes.

4.4.3 Modifying the Virtual File System Attribute

You may be able to improve Oracle 8.1.7.x/9*i* database performance by tuning the following (`vfs`) subsystem `fifo_do_adaptive` attribute.

The `fifo_do_adaptive` attribute specifies a value that enables (1) or disables (0) the pipe code that attempts to do batch writes to a pipe and deliver the data in a single call to a reader. The `fifo_do_adaptive` attribute is one of the tunables where the default value may not be appropriate if the system is running as a database server.

The default setting of 1 enables alternate algorithms in the FIFO routines. This will create an optimal working set size and will perform fewer data transfer operations, but of a larger size. The default works reasonably well for applications that perform data transfers of a uniform or near-uniform size. The default does not work so well for *some* applications that perform data transfers of a random size, particularly those that started out performing transfers such that the FIFO code determined an optimal transfer size.

The default value is ineffective for some applications in which the peer processes operate in sync; for example, `procA` transfers to `procB` and then waits for `procB`'s response. By disabling the `fifo_do_adaptive` parameter, performance for some applications degrades, and for other applications it improves. The performance change depends on how the pipes are used. We recommend setting this parameter to 0 in Oracle environments.

See sys_attrs_vfs(5) reference page for more information and see Chapter 3 for information about modifying kernel subsystem attributes.

4.4.4 Modifying Interprocess Communication Attributes

You may be able to improve Oracle 8.1.7.x/9*i* database performance by tuning the following ipc subsystem attributes:

- ssm_threshold (Section 4.4.4.1)
- shm_max (Section 4.4.4.2)
- shm_min (Section 4.4.4.3)
- shm_mni (Section 4.4.4.4)
- shm_seg (Section 4.4.4.5)

See sys_attrs_ipc(5) for more information and see Chapter 3 for information about modifying kernel subsystem attributes.

4.4.4.1 Modifying the System V Shared Regions

The ssm_threshold attribute specifies the minimum size, in bytes, of a System V shared region for the use of shared page tables. The ssm_thresold attribute controls which type of segmented shared memory (SSM) implementation is used.

The default value is 8 MB. Disable ssm_threshold if you are using large pages; that is, either rad_gh_regions or gh_chunks, (see Section 4.4.1.2). We recommend leaving ssm_threshold at its default value on non-GS series unless you are using rad_gh_regions or gh_chunks; in that case, set ssm_threshold to 0 (disabled).

4.4.4.2 Modifying the System V Maximum Size of Shared Memory Region

The shm_max attribute specifies the maximum size, in bytes, of a single System V shared memory region. Oracle concatenates multiple shared memory regions if the SGA is larger then the value configured for shm_max. The size for a single shared memory segment (SSM) could be larger than 2 GB. However, applications using shared memory on the same system may have problems with shared memory segments larger than 2 GB.

To avoid compatibility issues, we recommend setting the maximum size for an individual shared memory segment to 2 GB. The recommended value is 2 GB minus 8 MB (2139095040). If Oracle is the only application used on the system, you can increase the size of shm_max to 4 GB minus 16 MB (4278190080). The default value is 4,194,304 bytes (512 pages).

4.4.4.3 Modifying the System V Minimum Size of Shared Memory Region

The shm_min attributes specifies the minimum size, in bytes, of a single System V shared memory region.

The recommended value is one region and the default value is one region.

4.4.4.4 Modifying the Shared Memory Regions that Can be Used at One Time

The shm_mni attribute specifies the maximum number of shared memory regions that can be used on the system at one time.

The recommended value is 256 regions; the default value is 100 regions. The system rounds the number to the value associated with the next higher power of two, so the default value would actually be 128 regions.

4.4.4.5 Modifying the Shared Memory Regions that Can be Attached at One Time

The shm_seg attribute specifies the maximum number of System V shared memory regions that can be attached to a single process at one time.

The recommended value is 128 regions.

4.4.5 Modifying Internet Attributes

You may be able to improve Oracle 8.1.7.x/9*i* database performance by tuning the following inet subsystem attributes:

- udp_sendspace (Section 4.4.5.1)
- udp_recvspace (Section 4.4.5.2)
- ipport_unserreserved (Section 4.4.5.3)

See sys_attrs_inet(5) for more information and see Chapter 3 for information about modifying kernel subsystem attributes.

Also, for more information about tuning Internet servers see Chapter 6. For recommendations specific to GigaBit Ethernet performance, see:

http://www.tru64unix.compaq.com/docs/best_practices/BP_GIGABIT/TITLE.HTM.

4.4.5.1 Modifying the Send Buffer Size for the UDP Sockets

The udp_sendspace attribute specifies the default send buffer size, in bytes, for UDP sockets. If your application is using a Gigabit Ethernet or heavy network activity, consider setting the value of the udp_sendspace attribute to a higher value than the suggested default.

The recommended value is 65536 (bytes) or larger, depending on the size and number of user sessions and or queries.

4.4.5.2 Modifying the Receive Buffer Size for the UDP Sockets

The `udp_recvspace` attribute specifies the default receive buffer size, in bytes, for UDP sockets. If your application is using a Gigabit Ethernet or heavy network activity, consider setting the value of the `udp_recvspace` attribute to a higher value than the suggested default.

The recommended value is 65536 (bytes) or larger, depending on the size and number of user sessions and queries.

4.4.5.3 Modifying the Number of Times a System can make Outgoing Connections

The `ipport_userreserved` attribute specifies the number of times a system can simultaneously make outgoing connections to other systems.

The number of outgoing ports is the value of the `ipport_userreserved` attribute minus the value of the `ipport_userreserved_min` attribute. The default value is 5000 (bytes). Therefore, the default number of outgoing ports is 3976. The recommended value for large-scale Oracle installations is also the maximum value of 65535 (bytes).

4.4.6 Modifying Process Attributes

You may be able to improve Oracle 8.1.7.x/9*i* database performance by tuning the following `proc` subsystem attributes:

- `per_proc_stack_size` (Section 4.4.6.1)
- `max_per_proc_stack_size` (Section 4.4.6.2)
- `per_proc_data_size` (Section 4.4.6.3)
- `max_per_proc_data_size` (Section 4.4.6.4)
- `per_proc_address_space` (Section 4.4.6.5)
- `max_per_proc_address_space` (Section 4.4.6.6)
- `max_proc_per_user` (Section 4.4.6.7)
- `max_threads_per_user` (Section 4.4.6.8)
- `maxusers` (Section 4.4.6.9)

See `sys_attrs_proc`(5) for more information and see Chapter 3 for information about modifying kernel subsystem attributes.

4.4.6.1 Modifying the Per Process Stack Size

The per_proc_stack_size attribute specifies, in bytes, the per process stack size. The default value of 8 MB should be large enough for most Oracle environments. However, in very large installations and data warehouse type environments, the value should be increased.

The recommended value is 33,554,432 (32 MB).

4.4.6.2 Modifying the Maximum Size of the User Process Stack Size

The max_per_proc_stack_size attribute specifies the maximum size, in bytes, of a user process stack. The default value of 32 MB should be large enough for most Oracle environments. However, in very large installations and data warehouse type environments, the value should be increased.

The recommended value is 536,870,912 (512 MB). Depending on the Oracle environment, the maximum stack size can be increased to a maximum value of less than or equal to 896 MB. This limit is due to the fact that Oracle has a fixed PGA and fixed SGA at 0x38000000 and 0x58000000 for performance. If you use a larger value than 896 MB for this parameter, Oracle may corrupt the fixed SGA and PGA.

4.4.6.3 Modifying the Per Process Data Size

The per_proc_data_size attribute specifies, in bytes, the per process data size. Set the value to the amount of physical memory installed on your system. You can raise the value to a larger value than actual memory available. However, doing so would allow a single process to outgrow a system's main memory and cause extensive swapping and paging (see Section 12.5).

We recommend staying within the bounds of the available memory for the per_proc_data_size attribute. Never raise per_proc_data_size to a value larger than the physical memory available plus the configured swap space. The recommended value is the amount of physical memory installed, with a maximum value of 4,398,046,511,104 bytes.

4.4.6.4 Modifying the Maximum Size of the Per Process Data Size

The max_per_proc_data_size attribute specifies the maximum size, in bytes, of a data segment for each process. Set the value to the amount of physical memory installed on your system. You could raise the value to a larger value than actual memory available. However, doing so would allow a single process to outgrow a system's main memory and cause extensive swapping and paging (see Section 12.5).

We recommend staying within the bounds of the available memory for the max_per_proc_data_size attribute. Never raise max_per_proc_data_size to a value larger than the physical memory available plus the configured swap space. The recommended value is the amount of physical memory installed, with a maximum of 4,398,046,511,104 bytes.

4.4.6.5 Modifying the Per Process Address Size

The per_proc_address_space attribute specifies, in bytes, the per process address size. Set this value to the amount of physical memory installed on your system. You could raise the value to a larger value than actual memory available. However, this would allow a single process to outgrow a system's main memory and cause extensive swapping and paging (see Section 12.5).

We recommend staying within the bounds of the available memory for the per_proc_address_space attribute. Never raise per_proc_address_space to a value larger than the physical memory available, plus the configured swap space. The recommended value is the amount of physical memory installed, with a maximum value of 4,398,046,511,104 bytes.

4.4.6.6 Modifying the Maximum Per Process Address Size

The max_per_proc_address_space attribute specifies the maximum amount, in bytes, of user process address space. Set the value to the amount of physical memory installed on your system. You could raise the value to a larger value than actual memory available. However, doing so would allow a single process to outgrow a system's main memory and cause extensive swapping and paging (see Section 12.5).

We recommend staying within the bounds of the available memory for the max_per_proc_address_space attribute. Never raise max_per_proc_address_space to a value larger than the physical memory available plus the configured swap space. The recommended value is the amount of physical memory installed, with a maximum of 4,398,046,511,104 bytes.

4.4.6.7 Modifying the Maximum Number of Processes

The max_proc_per_user attribute specifies the maximum number of processes (tasks) that a user can create (the superuser is not affected). To disable the limits for the max_proc_per_user attribute, set the attribute value to 0.

The recommended value is 1024 (processes). If the application requires more than 1024 tasks per user, increase the value accordingly.

4.4.6.8 Modifying the Maximum Number of Threads

The max_threads_per_user attribute specifies the maximum limit of threads a user can create (the superuser is not affected). To disable the limits for the max_threads_per_user attribute, set the attribute value to 0.

The recommended value is 4096 threads. If the application requires more than 1024 tasks per user, increase the value accordingly.

4.4.6.9 Modifying the Space Allocated to System Tables

The maxusers attribute specifies the number of simultaneous users that a system can support without straining system resources. System algorithms use the maxusers value to size various system data structures and to determine the amount of space allocated to system tables, such as the system process table.

For ES40 class systems and higher, we recommend setting the attribute to a value of 8192 (users) or up to the maximum value of 16,384 (users). The default value is system-dependent.

4.4.7 Modifying the Real-Time Attribute

You may be able to improve Oracle 8.1.7.x/9*i* database performance by tuning the rt subsystem aio_task_max_num attribute.

The aio_task_max_num attribute specifies the limit that indirectly controls the number of AIO requests that can be wired in physical memory by restricting the amount of wired physical memory available for a specified number of tasks. One page of wired physical memory is available to the number of tasks specified by aio_task_max_num.

The recommended value should be greater than the maximum of either the DBWR I/O operations or the value of the DB_FILE_MULTI-BLOCK_READ_COUNT parameter, whichever is higher. The maximum number of DBWR I/O operations defaults to 8192 unless you specify the _DB_WRITER_MAX_WRITES initialization parameter. The default value is 102 (one page of wired memory per 102 tasks).

A simple formula for this tunable is:

```
(DB_WRITER_MAX_WRITES (default 8192) * DB_WRITER_PROCESSES)
+ (PARALLEL_MAX_SERVERS * DB_FILE_MULTIBLOCK_READ_COUNT) +
10.
```

See sys_attrs_rt(5) for more information and see Chapter 3 for information about modifying kernel subsystem attributes.

4.4.8 Modifying Reliable Datagram Attributes

You may be able to improve Oracle 8.1.7.x/9*i* database performance by tuning the following `rdg` subsystem attributes:

- `max_objs` (Section 4.4.8.1)
- `msg_size` (Section 4.4.8.2)
- `max_async_req` (Section 4.4.8.3)
- `max_sessions` (Section 4.4.8.4)
- `rdg_max_auto_msg_wires` (Section 4.4.8.5)

See `sys_attrs_rdg`(5) for more information and see Chapter 3 for information about modifying kernel subsystem attributes.

4.4.8.1 Modifying the Maximum Number of Objects in the RDG

The `max_objs` attribute specifies the maximum number of objects in the RDG endpoint and buffer list.

The recommended value is at least 5 times the number of Oracle processes per node, up to 10240 or the number of Oracle processes multiplied by 70.

4.4.8.2 Modifying the Maximum Size of the RDG Message

The `msg_size` attribute specifies the maximum size in bytes of an RDG message.

The recommended value is equal to or greater than the maximum value of the `DB_BLOCK_SIZE` parameter for the database. Oracle recommends a value of 32768 because Oracle9i supports different block sizes for each table space.

4.4.8.3 Modifying the Maximum Number of Messages in the RDG

The `max_async_req` attribute specifies the maximum number of asynchronous messages held in the RDG send and receive queues.

The recommended value is at least 100. A value of 256 might provide better performance.

4.4.8.4 Modifying the Maximum Number of Sessions within the RDG Table

The `max_async_req` attribute specifies the maximum number of sessions within any given RDG context table.

The recommended value is at least the number of Oracle processes plus two.

4.4.8.5 Modifying the Maximum Number of Pages Wired For Message Packets

The max_async_req attribute specifies the maximum number of pages automatically wired in memory for message packets.

We recommend setting the max_async_req attribute to 0.

4.4.9 Modifying the Memory Channel Attribute

You may be able to improve Oracle 8.1.7.x/9*i* database performance by tuning the rm subsystem rm_check_for_ipl attribute.

The rm_check_for_ipl attribute specifies the bitmask, indicating when the CPU priority level (spl) of the processor should be saved in a trace buffer.

We recommend setting this the attribute to 63, the default value.

See sys_attrs_rm(5) for more information and see Chapter 3 for information about modifying kernel subsystem attributes.

5

Tuning Network File Systems

The network file system (NFS) allows users to access files transparently across networks. The NFS supports a spectrum of network topologies, from small and simple networks to large and complex networks. The NFS shares the Unified Buffer Cache (UBC) with the virtual memory subsystem and local file systems.

NFS can put an extreme load on the network. Poor NFS performance is almost always a problem with the network infrastructure. Look for high counts of retransmitted messages on the NFS clients, network I/O errors, and routers that cannot maintain the load. Lost packets on the network can severely degrade NFS performance. Lost packets can be caused by a congested server, the corruption of packets during transmission (which can be caused by bad electrical connections, noisy environments, or noisy Ethernet interfaces), and routers that abandon forwarding attempts too quickly.

When evaluating NFS performance, remember that NFS does not perform well if any file-locking mechanisms are in use on an NFS file. The locks prevent the file from being cached on the client.

Improving performance on a system that is used only for serving NFS differs from tuning a system that is used for general timesharing, because an NFS server runs only a few small user-level programs, which consume few system resources. There is minimal paging and swapping activity, so memory resources should be focused on caching file system data.

File-system tuning is important for NFS because processing NFS requests consumes the majority of CPU and wall-clock time. Ideally, the UBC hit rate should be high. Increasing the UBC hit rate can require additional memory or a reduction in the size of other file-system caches (see Section 11.1.3). In general, file-system tuning will improve the performance of I/O-intensive user applications. In addition, a vnode must exist to keep file data. If you are using AdvFS, an access structure is also required to keep file data. See Chapter 11 for more information about file systems.

This chapter describes how to improve NFS performance. It offers various configuration guidelines and describes several monitoring tools, including the following topics:

- Monitoring NFS statistics (Section 5.1)

- Detecting poor performance (Section 5.2)
- Performance benefits and tradeoffs (Section 5.3)
- NFS configuration (Section 5.4)
- NFS retransmissions (Section 5.5)
- Tuning NFS servers (Section 5.6)
- Tuning NFS Clients (Section 5.7)

5.1 Monitoring NFS Statistics

This section provides references to utilities that you can use to gather NFS performance information. It is important that you gather statistics under a variety of conditions. Comparing sets of data will help you diagnose performance problems.

Table 5–1 describes the tools that you can use to detect poor NFS performance.

Table 5–1: Tools to Detect Poor NFS Performance

Tools	Description	Reference
`nfsstat`	Displays network file system statistics.	Section 2.4.3
`tcpdump`	Monitors and displays packet headers on a network interface.	Section 2.4.4
`netstat`	Displays network statistics.	Section 2.4.5
`ps axlm`	Displays idle I/O threads on a system.	Server side: Section 2.4.6 Client side: Section 2.4.7
`nfswatch`	Monitors all incoming network traffic to a NFS file server and divides it into several categories.	Section 2.4.8
`dbx print nfs_sv_active_hist`	Displays a histogram of the active NFS server threads.	Section 3.1
`dbx print nchstats`	Determines the namei cache hit rate.	Section 2.6
`dbx print bio_stats`	Determines the metadata buffer cache hit rate.	Section 3.1

5.2 Detecting Poor NFS Performance

Table 5–2 describes some poor NFS performance problems and possible solutions.

Table 5–2: Potential NFS Problems and Solutions

Problem	Solutions
NFS server threads busy.	Reconfigure the server to run more threads. See Section 2.4.6.
Memory resources are not focused on file-system caching.	Increase the amount of memory allocated to the UBC. See Section 11.1.3. If you are using AdvFS, increase the memory reserved for AdvFS access structures. See Section 11.1.5.
System resource allocation is not adequate.	Set the value of the `maxusers` attribute to the number of server NFS operations that are expected to occur each second. See Section 8.1.
UFS metadata buffer cache hit rate is low.	Increase the size of the metadata buffer cache. See Section 11.1.4. Increase the size of the namei cache. See Section 11.1.2.
CPU idle time is low.	Use UFS, instead of AdvFS. See Section 11.3.

5.3 Performance Benefits and Tradeoffs

Table 5–3 lists NFS configuration guidelines and performance benefits and trade-offs.

Table 5–3: NFS Tuning Guidelines

Benefit	Guideline	Tradeoff
Enable efficient I/O blocking operations	Configure the appropriate number of threads on an NFS server (Section 5.4.1)	None
Enable efficient I/O blocking operations	Configure the appropriate number of threads on the client system (Section 2.4.7)	None

Table 5–3: NFS Tuning Guidelines (cont.)

Benefit	Guideline	Tradeoff
Improve performance on slow or congested networks	Decrease network timeouts on the client system (Section 5.4.3)	Reduces the theoretical performance
Improve network performance for read-only file systems and enable clients to quickly detect changes	Modify cache timeout limits on the client system (Section 5.4.3)	Increases network traffic to server

The following sections describe these guidelines in more detail.

5.4 NFS Configuration

This section describes specific areas of the network file system (NFS) configuration. For more information about Network Configuration see the *Network Administration: Connections* guide.

5.4.1 Configuring Server Threads

The `nfsd` daemon runs on NFS servers to service NFS requests from client systems. The daemon spawns a number of server threads that process NFS requests from client systems. At least one server thread must be running for a machine to operate as a server. The number of threads determines the number of parallel operations and must be a multiple of 8.

To improve performance on frequently used NFS servers, configure either 16 or 32 threads, which provides the most efficient blocking for I/O operations. Configure 16 threads or more, up to a total of 128 combined UDP and TCP threads.

To monitor the number of UDP and TCP threads, use the following commands:

```
# ps axlm | grep -v grep | grep -c nfs_udp
# ps axlm | grep -v grep | grep -c nfs_tcp
```

The previous commands will display the number of sleeping or idle threads. If this number is repeatedly 0, additional `nfsd` threads should be configured. See Section 2.4.7 or `nfsd`(8) for more information.

5.4.2 Configuring Client Threads

Client systems use the `nfsiod` daemon to service asynchronous I/O operations, such as buffer cache read-ahead and delayed write operations. The `nfsiod` daemon spawns several I/O threads to service asynchronous

I/O requests to its server. The I/O threads improve performance of both NFS reads and writes.

The optimal number of I/O threads to run depends on many variables, such as how quickly the client is writing data, how many files will be accessed simultaneously, and the behavior of the NFS server. The number of threads must be a multiple of 8 minus 1 (for example, 7,15 ,31, and so forth is optimal).

NFS servers attempt to gather writes into complete UFS clusters before initiating I/O, and the number of threads (plus 1) is the number of writes that a client can have outstanding at one time. Having exactly 7 or 15 threads produces the most efficient blocking for I/O operations. If write gathering is enabled, and the client does not have any threads, you may experience a performance degradation. To disable write gathering, use the `dbx patch` command to set the `nfs_write_gather` kernel variable to 0. See Section 3.2 for information about the `dbx` command.

To display idle I/O threads on the client, use the following command:

```
# ps axlm | grep -v grep | grep -c nfsiod
```

If few threads are sleeping, you might improve NFS performance by increasing the number of threads. See Section 2.4.6 or `nfsiod`(8) for more information.

5.4.3 Modifying Cache Timeout Limits

For read-only file systems and slow network links, performance might improve by changing the cache timeout limits on NFS client systems. These timeouts affect how quickly you see updates to a file or directory that was modified by another host. If you are not sharing files with users on other hosts, including the server system, increasing these values will slightly improve performance and will reduce the amount of network traffic that you generate.

See `mount`(8) and the descriptions of the `acregmin`, `acregmax`, `acdirmin`, `acdirmax`, `actimeo` options for more information.

5.5 NFS Retransmissions

Excessive retransmissions can cause poor performance because the client must wait for the server to respond before it retransmits a request. Excessive retransmissions can be caused by the following problems:

- Overloaded servers that drop packets due to insufficient buffering
- Inadequate Ethernet transceivers that cause packets to be dropped under busy conditions

- Physical network errors, such as those caused by a noisy coaxial cable

Use the `nfsstat -c` command to measure the NFS retransmission rate on client machines. You can then determine the rate of retransmissions. See `nfsstat`(8) for more information.

The average NFS response time to a client request under a low to medium load is approximately 15 milliseconds. Most clients retransmit a request after approximately 1 second. If a 10 percent reduction in performance is acceptable, then a 1.5-millisecond increase in response time is an acceptable limit. This reduction gives an acceptable NFS retransmission rate of 0.15 percent. The calculation is as follows:

```
  .0015 sec/request
----------------------  =  0.0015 retransmission/request
1.0 sec/retransmission
```

Because the worst-case NFS request (read or write 8 KB over the Ethernet) requires seven packets (one request and six fragmented replies), the error rate of the network must be less than 0.02 percent. The calculation is as follows:

```
  0.15 percent
---------  = 0.02 percent
    7
```

Use the `netstat -i` command to measure the network error rate. If this rate is unacceptably high, determine if an individual machine is generating an excessive number of errors. If the problem appears to be pervasive, analyze the cabling technology that is being used. For example, if you have difficulties with noisy nonstandard coaxial cable, you could switch to a twisted-pair Ethernet. See `netstat`(1) for more information.

5.5.1 Decreasing Network Timeouts

NFS does not perform well if it is used over slow network links, congested networks, or wide area networks (WANs). In particular, network timeouts on client systems can severely degrade NFS performance. This condition can be identified by using the `nfsstat` command and determining the ratio of timeouts to calls. If timeouts are more than 1 percent of the total calls, NFS performance may be severely degraded. See Section 2.4.3 for sample `nfsstat` output of timeout and call statistics.

You can also use the `netstat -s` command to verify the existence of a timeout problem. A nonzero value in the `fragments dropped after timeout` field in the `ip` section of the `netstat` output may indicate that the problem exists. See Section 2.4.5 for sample `netstat` command output.

If fragment drops are a problem on a client system, use the `mount` command with the `-rsize=1024` and `-wsize=1024` options to set the size of the NFS read and write buffers to 1 KB.

5.6 Tuning NFS Servers

Tru64 UNIX uses a buffer cache in memory to avoid disk operations whenever possible. This memory is effective in reducing the client waiting time for relatively slow disk I/O. It also makes disk I/O more efficient by allowing the staging and scheduling of disk operations.

You can improve performance by allowing the disk device driver to schedule several requests at a time to take advantage of the position of the disk arm. The total amount of disk I/O is reduced, because repeat requests may be found in the cache. If NFS read activity is high, then adding more memory to your server can improve server performance because the size of the buffer cache is a percentage of the size of memory.

Performance problems at the server make the system buffer cache inefficient when serving remote write requests. NFS uses a simple stateless protocol, which requires that each client request be complete and self-contained and that the server completely process each request before sending an acknowledgment back to the client. If the server crashes or if an acknowledgment is lost, the client retransmits its request to the server. Because of this, the following events occur:

- The server cannot acknowledge the client's request until data is safely written to stable storage.
- The client knows exactly how much modified data has been safely stored by the server.
- The server cannot cache modified data in volatile storage because the data may be lost if the server crashes.

In NFS Version 2, write operations are synchronous. When the server receives a write request, it must write the data and information needed to find it later before replying. Tru64 UNIX uses a technique called write gathering to reduce this I/O load, but the performance impact is still very high.

In NFS Version 3, write requests are usually asynchronous, which minimize the performance impact of write operations. When the server first receives a write request, it merely acknowledges receipt of the data. Later, the client will send a commit request, requesting the server to write any data that is still in the cache and reply when all data is on stable storage. The protocol includes a write verifier that allows the client to detect if the server crashed and rebooted between the write and commit operations. If so, the client retransmits the uncommitted write requests to ensure that the server has the proper data.

You cannot use the system buffer cache to improve performance with NFS requests that modify data. If a server writes modified data only to volatile

memory, a server crash would jeopardize the data integrity. The client may assume that its data is safely stored, but if a crash occurs and the data was stored only in volatile memory, the data may be lost. Because a single server stores data for many clients, many clients can be affected. However, if modifications are always synchronously written to disk, data will not be lost and you can recover from server crashes.

Because NFS operations are synchronously committed to disk, a server can survive system failures since data integrity is ensured. However, performance is degraded because these operations take place at disk speeds and not at the memory speeds available to cachable operations. In addition, because these operations are processed serially, there is no opportunity to optimize the scheduling of the disk arm. Modifications to the cache are written synchronously to disk, so there is no opportunity to decrease write-disk traffic.

NFS servers run only a few small user-level programs, which consume few system resources. File-system tuning is important because processing NFS requests consumes the majority of CPU and wall-clock time. See Chapter 11 for information on file-system tuning.

In addition, if you are running NFS over Transmission Control Protocol (TCP), tuning TCP may improve performance if there are many active clients. See Section 10.2 for information on network subsystem tuning. If you are running NFS over User Datagram Protocol (UDP), network subsystem tuning is not normally needed.

Follow the guidelines in Table 5–4 to help you tune a system that is only serving NFS.

Table 5–4: NFS Server Tuning Guidelines

Guideline	Reference
Set the value of the `maxusers` attribute to the number of server NFS operations that are expected to occur each second.	Section 8.1
Increase the size of the namei cache.	Section 11.1.2
Increase the memory reserved for AdvFS access structures, if you are using AdvFS.	Section 11.1.5
Increase the size of the metadata buffer cache, if you are using UFS.	Section 11.1.4

5.6.1 Modifying NFS Server Side Attributes

You may be able to improve network file system server performance by tuning the following network file system (`nfs`) subsystem attributes:

- Write gathering (Section 5.6.1.1):
  - `nfs_write_gather` (Version 2.0)
  - `nfs_ufs_lbolt` (Version 2.0)
  - `nfs3_write_gather` (Version 3.0)
  - `nfs3_ufs_lbolt` (Version 3.0)
- Specifying the amount of time the server will delay the write (Section 5.6.1.2):
  - `nfs_slow_ticks`
  - `nfs_fast_ticks`
  - `nfs_unkn__ticks`
- Increasing the NFS send and receive buffer size (Section 5.6.1.3):
  - `nfs_tcpsendspace`
  - `nfs_tcprecvspace`

Note

Parameters for the `nfs` kernel subsystem are accessible only by using `dbx`; there are no comparable system attributes accessible through the `/sbin/sysconfig` command or the `dxkerneltuner` GUI. See Section 3.2 for more information about `dbx`.

See `sys_attrs_inet`(5) for more information and see Chapter 3 for information about modifying kernel subsystem attributes.

5.6.1.1 Write Gathering

Write gathering can improve the server capacity as it postpones disk writes of data requests by the client. Updating metadata is done less frequently compared to every time a write request is made. Write gathering provides a small amount of latency into the request processing cycle, waiting for more write requests to the same disk blocks to arrive at the server. However, the overall benefit of freeing up CPU cycles on the server outweighs the necessary overhead in most situations.

Write gathering also improves bandwidths as fewer, larger disk writes are completed; for example, there are fewer seeks and missed rotations. Some NFS V3 clients support asynchronous writes, but the benefit from server

write gathering is not as apparent. However, clients that do not support asynchronous writes, such as NFS V2 and some NFS V3 clients, need to do synchronous writes during recovery, which leaves write gathering turned on. This can improve system performance.

By default, write gathering is turned on. To receive the best results using this feature, tuning `nfsiods` on the clients can help improve the scalability of large servers scale.

There are `nfs` variables that are not applicable if `nfs_write_gather` is off. You can turn `nfs_ufs_lbolt` on or off, only if `nfs_write_gather` is turned on. The following conditions are the same for NFS V2 and NFS V3.

You can modify the variables under the following conditions:

1. If `nfs_write_gather` is on (default) → If `nfs_ufs_lbolt` is on → You can specify the time the server will delay the write (see Section 5.6.1.2).
2. If `nfs_write_gather` is on (default) → If `nfs_ufs_lbolt` is off → Modifying `nfs_*_ticks` does not have any effect.
3. If `nfs_write_gather` is off → If `nfs_ufs_lbolt` is off → Modifying `nfs_*_ticks` does not have any effect.

When serving dumb single threaded clients such as PCs or clients that do not support biods or clients that only emit writes infrequently, write gathering can slow down the clients as they wait for delayed replies from the server. This occurs because of the added overhead of latency added on the server side, whch delays writing. To improve a client's performance, disable `nfs_write_gather` and set it to 0 using `dbx -k /vmunix`. See Section 3.2 or Section 5.6.1.1.1 for more information about `dbx`.

5.6.1.1.1 Improving NFS Server Response Time to Client Write Requests

Changing the setting of the `nfs_ufs_lbolt` parameter to 0 might significantly improve NFS server response time to client write requests under either of the following conditions:

- The storage devices being written to on the NFS server include nonvolatile (battery-backed) cache.
- The NFS clients are predominantly systems (such as PCs) that always wait for a reply to one request before sending another request.

Setting `nfs_ufs_lbolt` has effect only when the NFS V2 protocol is being used. For NFS V2, the NFS server relies on a technique called write gathering to improve data throughput of synchronous write requests. One aspect of write gathering is to delay the return of a write reply to the client to include replies for any subsequent write requests, which might be received for the same file during a set interval. When data for all requests processed

during the delay interval are safely in storage, the server issues all the associated replies at the same time.

The period of time in which the server waits for more client requests is shorter than the time it takes to do a seek operation to disk but longer than it takes to flush data to the device's cache. Therefore, if the device cache is nonvolatile (the data is safely in storage before the transfer to media is complete), the time used by the server to wait for more requests is no longer efficient. Furthermore, the delay period degrades the performance of client systems that issue only one request as a time and then wait for a reply.

The following example shows how to use the `dbx assign` command to change the `nfs_ufs_lbolt` parameter in the running kernel, and the `dbx patch` command to ensure that the new setting is also made to the `/vmunix` file on disk:

```
# dbx -k /vmunix
dbx version 5.1
Type 'help' for help.

(dbx) print nfs_ufs_lbolt = 1
(dbx) assign nfs_ufs_lbolt = 0
(dbx) patch nfs_ufs_lbolt = 0
```

The `nfs_ufs_lbolt` parameter is not specific to using NFS V2 with UFS. Setting this parameter to 0 might also improve NFS V2 performance with AdvFS or the Cluster File System (CFS) as well. However, in a cluster environment, there can be a trade-off. The NFS server and the CFS server for NFS are not necessarily the same member system. If they are not and `nfs_ufs_lbolt` is set to 0, multiple replies to NFS write requests over TCP mounts are no longer batched in one RPC between the two servers in the cluster. In this case, the increase in the number of RPCs might degrade cluster performance.

Setting `nfs3_ufs_lbolt` to 0 will eliminate the same time interval as `nfs_ufs_lbolt` does but for requests using NFS V3 rather than NFS V2. NFS V3 relies far less on write gathering to handle client requests, and setting `nfs3_ufs_lbolt` to 0 is not likely to improve NFS V3 performance to any significant degree.

See Section 3.2 and Section 5.6.1.1.1 for more information about `dbx`.

5.6.1.2 Specifying the Amount of Time in Seconds the Server will Delay the Write

The `nfs` subsystem variables `nfs_slow_ticks`, `nfs_fast_ticks`, and `nfs_unkn_ticks` are specific to write gathering and are used to specify the amount of time, in seconds, the server will delay the write. Write gathering uses these variables to delay the write to give a larger window for more write requests coming for the same file.

The nfs variables are not valid if write gathering is off or if nfs_ufs_lbolt is turned off. See Section 5.6.1.1 for more information.

To identify what type of network card your system is using, enter one of the following commands:

- Trace the ifnet structure down to the desired interface and check the associated if_type using the following command:

  ```
  # dbx -k/vmunix
  ```

- Use the hwmgr command to display which network card type your system is using. For example:

  ```
  # hwmgr -get attr -cat network -a name -a sub_category 42:
  name = tu0
  sub_category = Ethernet
  ```

  The hwmgr utility is based on the value of if_type returned from the network drivers. There are three possible values for if_type:

  – IFT_ETHER (Ethernet)

  – IFT_FDDI (FDDI)

  – IFT_ISO88025 (Token ring and others).

  For more information about hwmgr, see Section 2.3.1.

To specify which of the three nfs variables to use, identify the network card type being used on your system as the media for NFS client/server communication and then match it with the specific variable in Table 5–5.

Table 5–5: Identifying Your Network Card Type

Network Card Type	Variable	Default Value (msec)
FDDI	nfs_fast_ticks	8
Ethernet	nfs_slow_ticks	5
Other	nfs_unkn_ticks	8

For newer and faster network cards, such as Gigabit Ethernet, decreasing the size of nfs_slow_ticks (IFT_ETHER) may result in increased performance.

5.6.1.3 Increasing the NFS Send and Receive Buffer Size

The nfs_tcpsendspace and nfs_tcprecvspace variables specify the NFS default send and receive buffer size for TCP sockets. If you are using a high-speed network adapter such as a Gigabit Ethernet, increasing these variables can improve system performance.

Use the following command to modify these variables:

```
# dbx -k/vmunix
```

The default values for `nfs_tcpsendspace` and `nfs_tcprecvspace` are 98304 bytes. For NFS V3, the default values are recommended for most network adapters and an I/O transfer size of 64 K. For NFS Version 2.0, the default values are recommended for an I/O transfer size of 8K.

Use `tcpdump` to determine if NFS on the remote system supports a TCP window size larger than 65536 bytes. The default size of the the TCP window is 65536 bytes. By default your supports RFC 1323, which allows you to set up larger window sizes through window scaling. However, if NFS on the remote system does not support RFC 1323, it will refuse the SYN packet sent at connection time.

Setting up a larger `nfs_tcpsendspace` window size on the server will speed up sending packets, increasing performance. On the client, if the client system also has Gigabit Ethernet, then the benefit would be the same.

See the *Network Programmer's Guide* for more information about window scaling.

5.7 Tuning NFS Clients

Adding disks or memory to a client can improve performance in two ways: by improving access time and by reducing the overall load on the server and network. A client can avoid network file system (NFS) performance problems for files that are not shared (such as root, swap, and temporary files) by using local disks for these files. For diskless clients, increased memory can make a big improvement in performance by allowing the client to swap and page less often. By adding local resources, the demands on the server and the network can be reduced.

While it is easy to improve client performance by adding memory or disks, these improvements may not be cost-effective because of the additional administrative tasks that are needed to maintain the operating system. For example, if you store valuable data on local disks, you must ensure that the disks are backed up. If the data is shared, you may also have to ensure that other systems have access. If you add resources to the server, the additional administrative costs are less than if you add the resources to the client.

The following sections describe how to improve `nfs` performance by modifying `nfs` subsystem attributes.

5.7.1 Modifying NFS Client Side Attributes

You may be able to improve NFS server performance by tuning the following `nfs` subsystem attributes:

- Improving read performance (Section 5.7.1.1):

  - nfs3_readaheads
  - nfs3_maxreadahead
- Controlling how long before the client will start transmitting (Section 5.7.1.2):
  - nfs3_jukebox_delay
- Directory name look up (Section 5.7.1.3 and Section 5.7.1.4) :
  - nfs_dnlc
  - nfs_nnc
- Specifying file consistency across NFS clients (Section 5.7.1.5):
  - nfs_cto
- Changing the NFS client behavior when fetching file attributes (Section 5.7.1.6):
  - nfs_quicker_attr
- Increasing the NFS send and receive buffer size (Section 5.6.1.3):
  - nfs_tcpsendspace
  - nfs_tcprecvspace

Note

Parameters for the nfs kernel subsystem are accessible only by using dbx; there are no comparable system attributes accessible throught the /sbin/sysconfig command or the dxkerneltuner GUI.

See sys_attrs_inet(5) for more information and see Chapter 3 for information about modifying kernel subsystem attributes.

5.7.1.1 Improving Read Performance

When the NFS Version 3.0 client completes a long sequential read or a partial block write and there are idled clients present, the NFS V3 client will attempt to read ahead. The nfs3_readahead variable specifies how many pages the client can read ahead, but not exceeding the maximum pages. The nfs3_maxreadahead variable species the maximum number of pages that the client can read ahead. The nfs3_readahead default value is 2 and the nfs3_maxreadahead default value is 8.

The read-ahead feature helps improve read performance. On systems with newer and faster network interfaces, tuning both variables as well as the number of running nfsiods helps saturate the network interface. This

maximizes the system hardware resource. Tuning this variable can double the read performance on newer gigabit network cards.

5.7.1.2 Controlling How Long Before the Client will Start Transmitting

The nfs3_jukebox_delay is the client's variable that controls how long, in seconds, before the client will start retransmitting again. For transactions on a busy server, nfs3_jukebox_delay can be increased to avoid unnecessary retransmission of client requests.The default value for nfs3_jukebox_delay is 10 seconds.

The nfs3_jukebox_delay variable is not related to any of the storage HSM mechanisms. The name is used to reflect the error message sent from the server, NFS3ERR_JUKEBOX. The term JUKEBOX reflects an NFS historic event and implies that the file is temporarily inaccessible. This allows the client to be aware of the server status and be able to make decisions to aggressively delay accessing the file rather than repeatedly retransmitting the request.

5.7.1.3 Directory Name Lookup Cache (DNLC)

The nfs_dnlc variable specifies the directory name lookup cache. By default, the client maintains a cache of results from recent file system directory lookup operations. As fewer server lookup requests are completed, client performance is improved.

To turn off the directory name lookup cache, specify the -noac option with the mount command. If -noac is not specified at mount time, you can turn off the nfs_dnlc variable to disable dnlc.

If the server is rapidly changing the files in the directory on the server, turning nfs_dnlc off can be useful. This will avoid some stale file handles by forcing opens to issue lookup calls.

5.7.1.4 Negative Name Cache Lookups (NNC)

The nfs_nnc variable specifies the negative name cache. In client lookup operations, when cache lookup fails, the client also maintains negative name cache to accommodate the failed vfs layer caching and to further eliminate unnecessary duplicate server lookups over the wire.

By default, nfs_nnc is turned on. For applications where minimal or no NFS directory lookup is done, turning nfs_nnc off can improve application performance.

If the server is rapidly changing the files in the directory on the server, turning nfs_dnlc off can be useful. This can avoid some stale file handles by forcing opens to issue lookup calls.

5.7.1.5 Specifying File Consistency Across NFS Clients

The nfs_cto variable specifies the closed-to-open (CTO) process, when a file is closed and all modified data associated with the file is flushed to the server or when the file is open, the client sends a request to the server to validate the client's local caches. This behavior ensures a file's consistency across multiple NFS clients.

When the -nocto option is specified at mount time, the client does not perform the flush on close. This allows the possibility of differences among copies of the same file as stored on multiple clients. For example:

```
# mount -nocto fubar:/abc/local
```

By default, nfs_cto is turned on. The benefit of keeping the variable turned on is that it solves the inconsistency of a file being accessed by multiple clients. If the first client makes a write to the file and closes it, when the second client opens it, the data on the second client is guaranteed to be up-to-date.

The benefit of turning nfs_cto off is when access to a specific file system will be made from only one client. Turning nfs_cto off can improve performance.

The client checks the close-to-open consistency at mount time. First, the client checks the nfs_cto variable and checks the setting of the mount -nocto option. Then the client proceeds to do close-to-open consistency by checking if nfs_cto is on or if the mount -nocto option is not set.

If nfs_cto is turned off at mount time using the -nocto option, set nfs_cto using dbx -k and the client will do CTO again on the mounted file system.

5.7.1.6 Changing the NFS Client Behavior When Fetching File Attributes

Setting the nfs_quicker_attr variable to any value other than 0, changes the NFS client behavior when fetching file attributes, for example ls -l or stat(2). By default, NFS waits for the file I/O to finish and executes the fsync( ) function to get the most up-to-date attributes. However, this can lead to delays when writing a file over a faster-than-disk interface. In a controlled environoment, setting this variable will fetch the cached attributes.

Modifying the nfs_quicker_attr variable can be useful in a testing or debugging environment when you want to observe the progress of writing a large file by repeatedly fetching file attributes, for example, using ls -l.

6

Tuning Internet Servers

This chapter describes how to tune Tru64 UNIX to improve your Internet server performance. It offers various configuration guidelines, describes several monitoring tools, and suggests primary and advanced tuning recommendations, including the following:

- Improving internet server performance (Section 6.1)
- Primary tuning recommendations (Section 6.2)
- Advanced tuning recommendations (Section 6.3)

Not all recommendations apply to all configurations, and some provide only marginal performance improvements. Therefore, you must fully understand your configuration and workload, and then carefully read the documentation before applying any recommendation.

Note

Some attribute names have changed for Tru64 UNIX Version 5.0 and higher.

6.1 Improving Internet Server Performance

This section describes how to improve your Internet server performance. It offers various configuration guidelines and describes several monitoring tools, including:

- Configuring hardware (Section 6.1.1)
- Configuring memory and swap space (Section 6.1.2)
- Logging IP addresses (Section 6.1.3)
- Monitoring network statistics (Section 6.1.4)
- Monitoring socket statistics (Section 6.1.5)
- Monitoring virtual memory statistics (Section 6.1.6)
- Gathering configuration information (Section 6.1.7)

6.1.1 Configuring Hardware

The following hardware configuration guidelines can help to improve Internet server performance:

- Make sure you have the latest version of the firmware for your system, disks, adapters, and controllers.
- Ensure that you have sufficient memory and swap space to handle the workload. See Section 6.1.2 for more information.
- Use high-performance storage hardware, including disks, adapters, and controllers in your Internet server configuration.
- Use Logical Storage Manager (LSM) or hardware RAID storage configurations for high performance and high availability.
- Use write-back caches in hardware RAID configurations to significantly improve Internet server performance.
- Place the `/tmp` and `/var/tmp` directories on different file systems and, if possible, different disks. For optimal performance, place the directories on disks under control of a RAID controller with the write-back cache option enabled.

6.1.2 Configuring Memory and Swap Space

You must provide sufficient memory and swap space to handle the server workload. Insufficient memory resources and swap space will cause performance problems. To configure memory and swap space, follow these steps:

1. Determine how much physical memory your workload requires.
2. Choose a swap space allocation mode, either immediate or deferred.
3. Determine how much swap space you need.
4. Configure the swap space to efficiently distribute the disk I/O.

In addition to the memory needed for system and application operations, each connection to an Internet server requires memory resources for the following:

- Kernel socket structure
- Internet protocol control block (`inpcb`) structure
- TCP control block structure
- Any socket buffer space that is needed as packets arrive and are consumed

These memory resources total 1 KB for each connection endpoint (not including the socket buffer space), which means that you will need 10 MB of memory to accommodate 10,000 connections.

You must ensure that your server has enough memory to handle demanding peak loads. Configure ten times more memory than what the server requires on a busy day, so that you have sufficient memory to handle occasional spikes of activity.

There are no limitations on a server's ability to handle millions of TCP connections if memory resources are available to service the connections. However, if you do not have sufficient memory, the server will reject new connection requests until enough existing connections are freed. Use the `netstat -m` command to monitor the memory that is currently being used by the network subsystem. See Section 6.1.4 for more information on the `netstat` command.

6.1.3 Logging IP Addresses

If your Internet server logs client host names, the application software may force the system to perform a reverse DNS lookup in order to obtain the client's host name. Reverse DNS lookups are time-intensive and may cause performance problems on busy servers with many clients.

You can modify the Internet software to log client Internet Protocol (IP) addresses, instead of client host names, without losing any significant information. Logging IP addresses may significantly improve the efficiency of the Internet server.

Consult the documentation provided by your Internet server software vendor to determine how to disable the logging of client host names. For example, you can obtain information about modifying Apache HTTP Server software from the Apache HTTP Server documentation Web site at this URL:

http://httpd.apache.org/docs/

6.1.4 Monitoring Network Statistics

The `netstat` command displays network statistics, including information about network routes and active sockets for each protocol. The command also displays cumulative statistics for network interfaces, including the number of incoming and outgoing packets and packet collisions, information about memory used for network operations, and statistics related to IP, ICMP, TCP, and UDP protocol layers.

Table 6–1 lists the `netstat` commands you can use to check network statistics.

Table 6–1: Tools for Monitoring Network Statistics

Tools	Description	Reference
`netstat -i`	Displays excessive amounts of input errors (`Ierrs`), output errors (`Oerrs`), or collisions (`Coll`), this may indicate a network problem.	Section 2.4.5.1
`netstat -is`	Checks for network device driver errors.	Section 2.4.5.2
`netstat -m`	Determines if the network is using an excessive amount of memory in proportion to the total amount of memory installed in the system.	Section 2.4.5.3
`netstat -an`	Determines the state of your existing network connections.	Section 2.4.5.4
`netstat -p ip`	Checks for bad checksums, length problems, excessive redirects, and packets lost because of resource problems.	Section 2.4.5.5
`netstat -p tcp`	Checks for retransmissions, out-of-order packets, and bad checksums.	Section 2.4.5.6
`netstat -p udp`	Checks for bad checksums and full sockets.	Section 2.4.5.6
`netstat -rs`	Displays routing statistics.	Section 2.4.5.7
`netstat -s`	Displays statistics related to IP, ICMP, IGMP,TCP, and UDP protocol layers.	Section 2.4.5.8
`sysconfig -q socket`	Displays the current attribute values. If the values show the queues are overflowing, you many need to increase the socket listen queue limit.	Section 6.1.5
`vmstat`	Displays data on virtual memory usage.	Section 6.1.6

See `netstat`(1) for more information.

6.1.5 Monitoring Socket Statistics

Three `socket` subsystem attributes monitor socket listen queue events:

- The `sobacklog_hiwat` attribute counts the maximum number of pending requests to any server socket.
- The `sobacklog_drops` attribute counts the number of times the system dropped a received SYN packet, because the number of queued SYN_RCVD connections for a socket equaled the socket's backlog limit.
- The `somaxconn_drops` attribute counts the number of times the system dropped a received SYN packet, because the number of queued SYN_RCVD connections for the socket equaled the upper limit on the backlog length (`somaxconn` attribute).

The initial value of these attributes at boot time is 0. Use the `sysconfig -q` socket command to display the current attribute values. If the values show that the queues are overflowing, you may need to increase the socket listen queue limit. For example:

```
# sysconfig -q socket
socket:
pftimerbindcpu = 0
sbcompress_threshold = 0
sb_max = 1048576
sobacklog_drops = 0
sobacklog_hiwat = 21
somaxconn = 65535
somaxconn_drops = 0
sominconn = 65535
mbuf_ext_lock_count = 64
umc_min_len = 1024
umc = 0
```

We recommend that the value of the `sominconn` attribute equal the value of the `somaxconn` attribute. If so, the value of `somaxconn_drops` will have the same value as `sobacklog_drops`.

However, if the value of the `sominconn` attribute is 0 (the default), and if one or more server applications uses an inadequate value for the backlog argument to its `listen` system call, the value of `sobacklog_drops` may increase at a rate that is faster than the rate at which the `somaxconn_drops` counter increases. If this occurs, you may want to increase the value of the `sominconn` attribute. See Section 6.2.3.2 for more information on the `sominconn` attribute.

6.1.6 Monitoring Virtual Memory Statistics

The `vmstat` command provides data on virtual memory usage. This may help you determine if a system is paging excessively, which can degrade Internet server performance. For example:

```
# vmstat 1
Virtual Memory Statistics: (pagesize = 8192)
procs    memory         pages                            intr        cpu
r  w  u  act  free wire fault cow zero react pin pout  in  sy  cs  us  sy  id
7 526 59 80K  758  45K  402M  94M 132M   1M  74M 139K 757 42K  1K  38  14  48
7 526 59 81K  278  45K   939   15  896    0   11    0 824  2K  1K  85  11   4
6 528 59 81K  285  45K   595   67  411    0   10    0 983  5K  2K  81  17   2
7 526 59 81K  353  45K   560   31  446    0   17    0 781  2K  1K  87  10   3
7 526 59 81K  353  45K   406    0  406    0    0    0  1K  4K  2K  85  13   2
7 527 59 81K  288  45K   406    0  406    0    0    0  1K  7K  4K  81  18   1
9 524 59 81K  350  45K   640   72  420    0   13    0 999  3K  2K  85  13   2
 .
 .
 .
```

The values in the `memory` fields are specified in 8-KB pages. Check the size of the free page list (`free`). Compare the number of free pages to the values for the active pages (`act`) and the wired pages (`wire`). The sum of the free, active, and wired pages should be close to the amount of physical memory in your system. Although the value for `free` should be small, if the value is consistently small (less than 128 pages) and accompanied by excessive paging and swapping, you may have a physical memory shortage.

Also, examine the pageout (`pout`) field. If the number of pageouts is consistently high, you may have insufficient memory. You also may have insufficient swap space or your swap space may be inefficiently configured. Use the `swapon -s` command to display your swap device configuration, and use the `iostat` command to determine which swap disk is being used the most.

See `vmstat`(1), `swapon`(8), and `iostat`(1) for more information.

6.1.7 Gathering Configuration Information

The `sys_check` script is a `ksh` script that gathers configuration information and formats this information into an HTML file. It warns you if it detects configuration problems, checks your kernel subsystem attribute settings, and provides attribute tuning recommendations. See Section 2.3.3 for more information.

Be sure to use the latest version of `sys_check`. You can obtain this from:

http://www.tru64unix.compaq.com/sys_check/sys_check.html

6.2 Primary Tuning Recommendations

There are many kernel subsystem attributes that affect Internet server performance. Internet servers include Web servers, ftp servers, mail servers and relays, proxy servers, caching servers, gateway systems, and firewall systems. This section offers primary tuning recommendations for some of the attributes for the following subsystems:

- Internet (Section 6.2.1)

- Process (Section 6.2.2)
- Socket (Section 6.2.3)

Note

Some kernel subsystem attributes enable you to modify their value and apply the value to a running system. Other attributes require you to reboot the system to use a new value. See Section 3.3.1 to determine if an attribute can be tuned at run time.

The primary tuning recommendations provide the best performance improvement for most Internet server configurations. If performance is still deficient after applying these recommendations, you may be able to improve performance by modifying additional kernel subsystem attributes, as discussed in Section 6.3.

You can also use the Compaq Continuous Profiling Infrastructure (CPI, formerly known as DCPI) tool to obtain detailed information about system components that heavily utilize CPU cycles. CPI is offered as an Advanced Developement Kit. See to the following Web site for more information:

`http://www.tru64unix.compaq.com/dcpi`

6.2.1 Modifying Internet Attributes

You may be able to improve Internet server performance by tuning the following Internet `inet` subsystem attributes:

- `tcbhashsize` (Section 6.2.1.1)
- `pmtu_enabled` (Section 6.2.1.2)
- `ipport_userreserved` (Section 6.2.1.3)

See `sys_attrs_inet`(5) reference page for more information and see Chapter 3 for information about modifying kernel subsystem attributes.

6.2.1.1 Increasing the Size of the TCP Hash Table

The `tcbhashsize` attribute specifies the number of buckets in the Transmission Control Protocol (TCP) `inpcb` hash table. The kernel must look up the connection block for every TCP packet it receives; therefore, increasing the size of the table can speed up the search and improve performance.

However, increasing the size of the hash table will cause a slight increase in wired memory. It can also cause a bottleneck at the TCP hash table in SMP systems.

The default value is 512 buckets. The recommended value is 16384.

6.2.1.2 Disabling PMTU Discovery

Packets transmitted between servers are divided into equal-sized units to facilitate the transmission of the data over routers and small-packet networks, such as Ethernet networks.

When the `pmtu_enabled` attribute is enabled, the operating system determines the largest common path maximum transmission unit (PMTU) value between servers and uses it as the unit size. A routing table entry is also created for each client network that attempts to connect to the server.

If you have a poorly performing Internet server that handles mainly remote traffic and the routing table increases to more than 1000 entries, disabling the PMTU discovery can decrease the size of the routing table, which may improve server efficiency. However, if a server handles mainly local traffic and only some remote traffic, disabling PMTU discovery can degrade bandwidth. Use the `netstat -r` command to display the contents of the routing table.

The default value is 1 (PMTU enabled). The recommended value is 0 (PMTU disabled).

6.2.1.3 Increasing the Number of Outgoing Connection Ports

When a TCP or UDP application creates an outgoing connection, the kernel dynamically allocates a nonreserved port number for each connection.

The kernel selects the port number from a range of values between `ipport_userreserved_min` and `ipport_userreserved`.

Using the default attribute values, the range of outgoing ports begins at port 1024 and ends at port 5000, and the number of simultaneous outgoing connections is limited to 3976 (5000 minus 1024).

If you have a proxy server, caching server, gateway system, or firewall system with a load of more than 4000 simultaneous connections, you can modify the value of the `ipport_userreserved` attribute. The default value is 5000, which is the minimum value. The recommended value is 65535, which is the maximum value. Do not specify a value that is greater than 65535 or lower than 5000.

6.2.2 Modifying Process Attributes

You may be able to improve Internet server performance by tuning the following process `proc` subsystem attributes:

- `maxusers` (Section 6.2.2.1)

- max_proc_per_user (Section 6.2.2.2)
- max_threads_per_user (Section 6.2.2.3)
- max_per_proc_data_size (Section 6.2.2.4)
- max_per_proc_address_space (Section 6.2.2.5)

These attributes set limits on system resources. If your Internet server appears to be reaching the resource limits, you may want to increase the value of one or more of these attributes. However, increasing the value of these attributes will allow the system to consume more memory.

See sys_attrs_proc(5) reference page for more information and see Chapter 3 for information about modifying kernel subsystem attributes.

6.2.2.1 Increasing the Size of System Tables and Data Structures

System algorithms use the maxusers attribute to size various system data structures and system tables. Increasing the value of maxusers provides more system resources to processes. However, this will increase the amount of wired memory.

If your system experiences a lack of resources (for example, Out of processes, No more processes, or pid table is full messages) and you have enough memory, increase the value of the maxusers attribute.

To determine an appropriate value for the maxusers attribute, you can double the default value until you improve performance. For example, if you have up to 1 GB of memory, increase the value of the maxusers attribute to 512. If you have up to 2 GB, increase the value to 1024. If you have an Internet, Web, proxy, caching, firewall, or gateway server, increase the value of the maxusers attribute to 2048.

The default value varies from 16 to 2048, depending on the amount of physical memory in the system. It is not recommended that you increase the value to more than 2048.

System administrators can change the maxusers attribute with the following command:

```
# sysconfig -r proc maxusers=N
```

The value *N* is the desired new value. This command triggers the automatic expansion of the pid table. The resizing of other system tables is not performed until you specify a new value for the maxusers attribute in the /etc/sysconfigtab file and reboot the system.

6.2.2.2 Increasing the Number of Processes per User

The max_proc_per_user attribute specifies the maximum number of processes that can be allocated at any one time to each user, except superuser.

If your system experiences a lack of processes, increase the value of this attribute. If you have a multiprocess Internet server (for example, a server running IPlanet, Apache, CERN, or Zeus), you also may want to increase the value of this attribute. Note that increasing its value increases the amount of wired memory.

The default value is 64. The recommended value is 2000. The value you choose must not be more than the maximum number of processes that can be started by your system. For Internet servers, these processes include CGI processes. If you specify a value of 0 for this attribute, there is no limit on the number of processes per user.

6.2.2.3 Increasing the Number of Threads per User

The `max_threads_per_user` attribute specifies the maximum number of threads that can be allocated at any one time to each user, except superuser.

If your system experiences a lack of threads, increase the value of this attribute. If you have a multithreaded Internet server (for example, a server running Netscape FastTrack or Netscape Enterprise), you may want to increase the value this attribute.

The default value is 256. The recommended value is 4096. The value must not be more than the maximum number of threads that can be started by your system.

6.2.2.4 Increasing the User Process Data Segment Size Limits

The `max_per_proc_data_size` attribute specifies the maximum limit of data segment sizes. Some large programs and large-memory processes may not run unless you increase the values of this attribute. Increase the limits if you receive an `Out of process memory` message.

The default value is 1073741824 (1 GB). The recommended value is 10737418240 (10 GB). If your system has more than 10 GB of memory, you can further increase this value.

6.2.2.5 Increasing the User Process Address Space Limits

The `max_per_proc_address_space` attribute specifies the maximum limit of user process address space (number of bytes of virtual memory). Some large programs and large-memory processes may not run unless you increase the value of this attribute. However, increasing the address space limits will cause a small increase in memory consumption.

The default value is 4294967296 (4 GB) for systems running Tru64 UNIX Version 5.0 or higher.

The recommended value is 10737418240 (10 GB). If your system has more than 10 GB of memory, you can further increase this value.

6.2.3 Modifying Socket Attributes

You may be able to improve Internet server performance by tuning the following socket attributes:

- somaxconn (Section 6.2.3.1)
- sominconn (Section 6.2.3.2)
- sbcompress_threshold (Section 6.2.3.3)

See sys_attrs(5) reference page for more information and see Chapter 3 for information about modifying kernel subsystem attributes.

6.2.3.1 Increasing the Maximum Number of Pending TCP Connections

The somaxconn attribute specifies the maximum number of pending TCP connections (the socket listen queue limit) for each server socket (for example, for the HTTP server socket). Pending TCP connections can be caused by lost packets in the Internet or denial of service attacks. Busy Internet servers often experience large numbers of pending connections. If the listen queue connection limit is too small, incoming connect requests may be dropped.

The default value is 1024. The recommended value is 65535, which is the maximum value. Do not specify a value that is higher than the maximum value because this can cause unpredictable behavior.

6.2.3.2 Increasing the Minimum Number of Pending TCP Connections

The sominconn attribute specifies the minimum number of pending TCP connections (backlog) for each server socket. The attribute controls the maximum number of SYN packets that the system can handle simultaneously before additional requests are discarded. Network performance can degrade if a client saturates a socket listen queue with erroneous TCP SYN packets, which blocks other users from the queue.

The value of the sominconn attribute overrides the application-specific backlog value, which may be set too low for some server software. If you do not have your application source code, use the sominconn attribute to set a pending-connection limit that is appropriate for your application.

The default value is 0. The recommended value is 65535, which is the maximum value. It is recommended that the value of the sominconn attribute be the same as the value of the somaxconn attribute. See Section 6.2.3.1 for more information in the somaxconn attribute.

6.2.3.3 Enabling the mbuf Cluster Compression

The sbcompress_threshold attribute controls whether mbuf clusters are compressed at the socket layer. By default, mbuf clusters are not compressed, which can cause proxy servers and caching servers to consume all the available mbuf clusters. This problem is more likely to occur if you are using FDDI instead of Ethernet. See Section 2.4.5.3 for information about monitoring mbuf clustering.

To enable mbuf cluster compression, modify the sbcompress_threshold attribute and specify a value. Packets will be copied into the existing mbuf clusters if the packet size is less than this value.

The default value is 0 (mbuf compression is disabled). If you have a proxy server, caching server, gateway system, or firewall system, the recommended value is 600 bytes.

6.3 Advanced Tuning Recommendations

This section offers advanced tuning recommendations for some of the attributes for the following subsystems:

- Generic (Section 6.3.1)
- Internet (Section 6.3.2)
- Network (Section 6.3.3)
- Socket (Section 6.3.4)
- Virtual memory (Section 6.3.5)

These recommendations are appropriate only for systems that are primarily used as Internet servers and are configured with sufficient physical memory. Using a recommended attribute value in a non-Internet server may cause degradation in system performance.

Because Internet server configurations differ and a recommended value may not provide optimal performance for all configurations, be careful when modifying attributes. Read the attribute descriptions and determine which values are appropriate for your configuration. If modifying an attribute does not improve performance, you may want to return to the default value.

6.3.1 Modifying Generic Attributes

You may be able to improve Internet server performance by tuning the kmemreserve_percent generic (generic) subsystem attribute. This attribute increases the percentage of physical memory reserved for kernel memory allocations that are less than or equal to the page size (8 KB). Increasing the value of kmemreserve_percent improves network

throughput by reducing the number of packets that are dropped while the system is under a heavy network load. However, increasing this value consumes memory.

You may want to increase the value of the `kmemreserve_percent` attribute if the output of the `netstat` command shows dropped packets, or if the output of the `vmstat -M` command shows dropped packets under the `fail_nowait` heading. This may occur under a heavy network load.

The default value is 0 (the percentage of reserved physical memory will be the smallest of 0.4 percent of available memory and 256 KB). Increase the value (up to a maximum of 75) by small increments until the output of the `vmstat -M` command shows no entries under the `fail_nowait` heading.

6.3.2 Modifying Internet Attributes

You may be able to improve Internet server performance by tuning the following Internet `inet` subsystem attributes:

- `tcbhashnum` (Section 6.3.2.1)
- `inifaddr_hsize` (Section 6.3.2.2)
- `tcp_keepinit` (Section 6.3.2.3)
- `tcp_rexmit_interval_min` (Section 6.3.2.4)
- `tcp_keepalive_default` (Section 6.3.2.5)
- `tcp_msl` (Section 6.3.2.6)
- `ipport_userreserved_min` (Section 6.3.2.7)
- `ipqs` (Section 6.3.2.8)
- `ipqmaxlen` (Section 6.3.2.9)

See `sys_attrs_inet`(5) and Chapter 3 for information about modifying kernel subsystem attributes.

6.3.2.1 Increasing the Number of TCP Hash Table

The `tcbhashnum` attribute specifies the number of TCP hash tables. Increasing the number of hash tables distributes the load and may improve performance. However, this will slightly increase the amount of wired memory in the system.

The default value is 1 hash table, which is the minimum value. For busy Internet server SMP systems, the recommended value is 16. The maximum value is 64.

If you increase the number of hash tables, decrease the size of the hash table. See Section 6.2.1.1 for more information. In addition, it is recommended

that you make the value of this attribute the same as the value of the ipqs attribute. See Section 6.3.2.8 for more information on the ipqs attribute.

6.3.2.2 Increasing the Number of Hash Buckets

The inifaddr_hsize attribute specifies the number of hash buckets in the kernel interface alias table (in_ifaddr).

If a system is used to serve many different server domain names, each of which are bound to a unique IP address, the code that matches arriving packets to the right server address uses the hash table to speed lookup operations for the IP addresses. These addresses are usually set using the ifconfig alias or ifconfig aliaslist command. Increasing the number of hash buckets in the table can improve performance on systems that use large numbers of IP alias addresses.

The default value is 32 hash buckets. For most Internet servers that do not use interface IP aliases or if you are using less than 250 aliases, the recommended value is 32. If you are using more than 500 interface IP aliases, the recommended value is 512, which is the maximum value.

For the best performance, the value of this attribute must be rounded down to the nearest power of 2.

6.3.2.3 Modifying the TCP Partial Connection Timeout Limit

The tcp_keepinit attribute specifies the amount of time that a partially established TCP connection remains on the socket listen queue before it times out. The value of the attribute is in units of 0.5 seconds. Partial connections consume socket listen queue slots and fill the queue with connections in the SYN_RCVD state.

The default value is 150 units (75 seconds). You do not need to modify the TCP partial-connection timeout limit unless the value of the somaxconn_drops attribute often increases. See Section 6.1.5 for more information on the event counter.

If your socket queue limit is set to the maximum value, the default value of this attribute is usually adequate. If the somaxconn_drops attribute often increases, and increasing the socket queue limit does not prevent the listen queue from filling up, you can decrease the value of this attribute to make partial connections to time out sooner.

In addition, network performance can degrade if a client overfills a socket listen queue with TCP SYN packets, which blocks other users from the queue. To eliminate this problem, increase the socket listen queue limit to its maximum value. If the system continues to drop SYN packets, decrease the value of this attribute to 30 (15 seconds). Monitor the values of the

sobacklog_drops and somaxconn_drops event counters to determine if the system is dropping packets.

Do not set the value of this attribute too low, because you may prematurely break connections with clients on slow network paths or network paths that lose many packets. Do not set the value to less than 20 units (10 seconds).

6.3.2.4 Decreasing the Rate of TCP Retransmissions

The tcp_rexmit_interval_min attribute specifies the minimum amount of time between the first TCP retransmission. For some wide area networks (WANs), the default value may be too small and premature retransmission timeouts may occur, which cause duplicate transmission of packets and the erroneous invocation of the TCP congestion-avoidance algorithms.

You can increase the value of this attribute to slow the rate of TCP retransmissions, which decreases congestion and improves performance.

The default value is 2 units (1 second). Not every connection needs a long retransmission time. Usually, the default value of this attribute is adequate. However, for some WANs, the default retransmission interval may be too small.

To check for retransmissions, use the netstat -p tcp command and examine the output for data packets retransmitted.

You can increase the value of this attribute to slow the rate of TCP retransmissions. The attribute is specified in units of 0.5 seconds.

Do not change the default value of this attribute unless you fully understand TCP algorithms. Do not specify a value that is less than 1 unit.

6.3.2.5 Enabling TCP Keepalive Functionality

Keepalive functionality enables the periodic transmission of messages on a connected socket to keep connections active and to time out inactive connections. Sockets that do not exit cleanly are cleaned up when the keepalive interval expires. If keepalive is not enabled, those sockets continue to exist until you reboot the system.

Applications enable keepalive for sockets by setting the setsockopt function's SO_KEEPALIVE option. The default value is 0 (keepalive is disabled). To enable keepalive for programs that do not set keepalive on their own, or if you do not have access to the application source code, set this attribute to 1. After you set the attribute, all new connections will have keepalive enabled; existing connections will continue to use the previous keepalive setting.

If you modify this attribute without rebooting the system, sockets that already exist will continue to use the old behavior until the applications are restarted.

If you enable keepalive, you can also configure the following TCP options for each socket:

- The `tcp_keepidle` attribute specifies the amount of idle time, in 0.5-second units, before sending a keepalive probe. The default value for this attribute is 2 hours.
- The `tcp_keepintvl` attribute specifies the amount of time, in 0.5-second units, between retransmission of keepalive probes. The default value for this attribute is 75 seconds.
- The `tcp_keepcnt` attribute specifies the maximum number of keepalive probes that are sent before the connection is dropped. The default value for this attribute is 8 probes.
- The `tcp_keepinit` attribute specifies the maximum amount of time, in 0.5-second units, before an initial connection attempt times out. The default value for this attribute is 75 seconds.

6.3.2.6 Increasing the TCP Connection Context Timeout Rate

The `tcp_msl` attribute determines the maximum lifetime of a TCP segment and the timeout value for the `TIME_WAIT` state. The TCP protocol includes a concept known as the Maximum Segment Lifetime (MSL). When a TCP connection enters the `TIME_WAIT` state, it must remain in this state for twice the value of the MSL, or else undetected data errors on future connections can occur.

You can decrease the value of this attribute to make the TCP connection context time out more quickly at the end of a connection. However, this will increase the chance of data corruption.

The default value is 60 units (30 seconds, which means that the TCP connection remains in `TIME_WAIT` state for 60 seconds or twice the value of the MSL). The value of this attribute is set in units of 0.5 seconds. The recommended value is the default value; if you use a different value, there is the potential for data corruption.

Although the TCP specifications specify an MSL of 120 seconds, most TCP implementations use a value that is less than 120. The *Internet FAQ Consortium* Web site offers more information. For RFC793, see the following URL:

http://www.faqs.org/rfcs/rfc793.html

For RFC1122, see the following URL:

http://www.faqs.org/rfcs/rfc1172.html

In some situations, the default timeout value for the TIME_WAIT state is too large, so reducing the value of this attribute frees connection resources sooner than the default behavior.

Do not reduce the value of this attribute unless you fully understand the design and behavior of your network and the TCP protocol.

6.3.2.7 Modifying the Range for Outgoing Connection Ports

When a TCP or UDP application creates an outgoing connection, the kernel dynamically allocates a nonreserved port number for each connection.

The kernel selects the port number from a range of values between ipport_userreserved_min and ipport_userreserved.

If your system requires a particular range of ports, you can modify the value of this attribute.

The default value is 1024. The maximum value is 65535. Do not specify a value for this attribute that is greater than 65535 or less than 1024.

6.3.2.8 Increasing the Number of IP Input Queues

For SMP systems, increasing the number of IP input queues can reduce lock contention at the input queue and distribute the load. The ipqs attribute specifies the number of IP input queues.

The default value is 1 queue, which is the minimum value. For busy Internet server SMP systems, the recommended value is 16. The maximum value is 64.

It is recommended that you make the value of this attribute the same as the value of the tcbhashnum attribute. See Section 6.2.1.1 for more information on the tcbhashnum attribute.

6.3.2.9 Increasing the Maximum Length of the IP Input Queue

If the network load is heavy, input packets may be dropped if the IP input queue becomes filled. The ipqmaxlen attribute specifies the maximum length, in bytes, of the IP input queue (ipintrq) before input packets are dropped.

If your system drops input packets, you may want to increase the value of the ipqmaxlen attribute. Check for dropped input packets by using dbx to examine the ipintrq kernel structure. For example:

```
# dbx -k /vmunix
(dbx) print ipintrq
struct {
     ifq_head = (nil)
     ifq_tail = (nil)
```

```
    ifq_len = 0
    ifq_maxlen = 512
    ifq_drops = 128
    ifq_slock = struct {
        sl_data = 0
        sl_info = 0
        sl_cpuid = 0
        sl_lifms = 0
     }
}
```

If the `ifq_drops` field is not 0, the system is dropping IP input packets.

The default value is 1024. The minimum value is the default value; the maximum value is 65535. If your system is dropping input packets, the recommended value is 2048. You may also want to increase the value of the `ifqmaxlen` attribute, which controls the output queue. See Section 6.3.3.1 for more information on the `ifqmaxlen` attribute.

6.3.3 Modifying Network Attributes

You may be able to improve Internet server performance by tuning the following Network `net` subsystem attributes:

- `ifqmaxlen` (Section 6.3.3.1)
- `screen_cachedepth` (Section 6.3.3.2)
- `screen_cachewidth` (Section 6.3.3.2)
- `screen_maxpend` (Section 6.3.3.3)

See `sys_attrs_net`(5) reference page for more information and see Chapter 3 for information about modifying kernel subsystem attributes.

6.3.3.1 Increasing the Number of Output Packets Before Packets are Dropped

If the network load is heavy, output packets may be dropped if the interface's output queue becomes filled. The `ifqmaxlen` attribute specifies the number of output packets that can be queued to a network adapter before packets are dropped.

You can use the `netstat -id` command to check for dropped output packets. If the command output shows a nonzero value in the `Drop` column for an interface, the system is dropping output packets and you may want to increase the value of this attribute.

The default value is 1024. The minimum value is the default value; the maximum value is 65535. If your system is dropping input packets, the recommended value is 2048.

6.3.3.2 Reducing Screening Cache Misses

If your machine is acting as a screening router, or a screening firewall running the `screend` facility, and has a high number of concurrent pass-through connections, you could be experiencing screening cache misses.

A screening cache miss can occur when the kernel screening table is trying to screen a packet that does not have an entry, based on address/port pairs and protocol. In that case, the table must queue the packet and the `screend` daemon must examine it. This can normally occur for the first packet of a connection, and can also occur if the cache is too small to hold many entries.

Check for screening cache misses by using `dbx` to examine the number of screening cache hits and misses. For example:

```
(dbx) p screen_cachemiss
616738
(dbx) p screen_cachehits
11080198
```

If the ratio of misses to hits is high, you may want to increase the values of the `screen_cachedepth` and `screen_cachewidth` attributes.

The default value for the `screen_cachedepth` attribute is 8, which is the minimum value. If you have high screening cache miss rates, the recommended value is 16, which is the maximum value.

The default value for the `screen_cachewidth` attribute is 8, which is the minimum value. If you have high screening cache miss rates, the recommended value is 2048, which is the maximum value.

It is recommended that you first increase `screen_cachewidth` before increasing `screen_cachedepth`. Also note that tuning these attributes will not necessarily reduce screening cache misses to 0. A reboot is required for the changes to take effect.

Increasing these values will cause a small increase in memory consumption.

6.3.3.3 Reducing the Screening Buffer Drops

If your machine is acting as a screening router, or a screening firewall running the `screend` facility, and is under heavy network load, you may be experiencing screening buffer drops.

You can use the `screenstat` command to view the current status. For example:

```
# /usr/sbin/screenstat
total packets screened: 11696910
total accepted: 11470734
total rejected: 225453
packets dropped:
        because buffer was full:        34723
```

```
        because user was out of sync:   0
        because too old:        0
total dropped:  34723
```

If the number of packets dropped `because buffer was full` is high, you may want to increase the value of the `screen_maxpend` attribute. The default value is 32, which is the minimum value. If you have a high screening buffer full value, the recommended value is 8192. The maximum value is 16384.

Increasing this value will cause a small increase in memory consumption. You must reboot the system to modify this attribute.

6.3.4 Modifying Socket Attributes

You may be able to improve Internet server performance by tuning the `sb_max` socket (`socket`) subsystem attribute. In addition, the `socket` subsystem attributes `sobacklog_hiwat`, `sobacklog_drops`, and `somaxconn_drops` track events related to socket listen queues. By monitoring these attributes, you can determine if the queues are overflowing. Section 6.1.5 discusses these attributes.

The `sb_max` attribute specifies the maximum size of a socket buffer. Increasing the maximum size of a socket buffer may improve performance if your applications can benefit from a large buffer size.

The default value is 1048576 bytes. If your applications require a socket buffer that is larger than the default value, increase the value of this attribute.

See `sys_attrs`(5) and Chapter 3 for information about modifying kernel subsystem attributes.

6.3.5 Modifying Virtual Memory Attributes

You may be able to improve Internet server performance by tuning `ubc_maxpercent`, `ubc_minpercent`, and `ubc_borrowpercent` virtual memory `vm` attributes.

Busy Internet servers usually consume a moderate amount of virtual memory and use a large set of files. Both processes and the Unified Buffer Cache (UBC), which caches file-system data, share the physical memory that is not wired by the kernel.

Too much memory allocated to the UBC can cause excessive paging and swapping, which may degrade overall system performance. However, an insufficient amount of memory allocated to the UBC can degrade file system performance.

The ubc_minpercent attribute specifies the minimum percentage of memory that only the UBC can utilize. The remaining memory is shared with processes. The ubc_maxpercent attribute specifies the maximum percentage of memory that the UBC can utilize. The ubc_borrowpercent attribute specifies the UBC borrowing threshold.

Between the value of the ubc_borrowpercent attribute and the value of the ubc_maxpercent attribute, the memory that is allocated to the UBC is considered borrowed from processes. When paging begins, these borrowed pages are reclaimed first, until the amount of memory allocated to the UBC decreases to the value of the ubc_borrowpercent attribute.

The default value for ubc_minpercent is 10 percent. The default value for ubc_maxpercent is 100 percent. The default value for ubc_borrowpercent is 20 percent. On a typical Internet server, the default value for each attribute is usually adequate. Also, if your disks are busy with file system I/O and the system has sufficient free pages, use the default values.

Use the vmstat command to display information about virtual memory, including the free page count.

If you have a low free page count, you may want to increase the memory available to processes by reducing the memory available to the UBC. You should attempt to keep in memory the working set of your processes, even if it increases the number of UBC misses.

You can reduce the default value of the ubc_maxpercent attribute in decrements of 10 percent.

Reducing the borrowed memory threshold by decreasing the value of the ubc_borrowpercent attribute may improve the system response time when memory is low. However, this may also reduce UBC performance.

7

Managing Application Performance

You may be able to improve overall Tru64 UNIX performance by improving application performance. This chapter describes application performance guidelines, see Table 7–1.

7.1 Improving Application Performance

Well-written applications use CPU, memory, and I/O resources efficiently. Table 7–1 describes some guidelines to improve application performance.

Table 7–1: Application Performance Improvement Guidelines

Guideline	Performance Benefit	Tradeoff
Install the latest operating system patches (Section 7.1.1)	Provides the latest optimizations	None
Use the latest version of the compiler (Section 7.1.2)	Provides the latest optimizations	None
Use parallelism (Section 7.1.3)	Improves SMP performance	None
Optimize applications (Section 7.1.4)	Generates more efficient code	None
Use shared libraries (Section 7.1.5)	Frees memory	May increase execution time
Reduce application memory requirements (Section 7.1.6)	Frees memory	Program may not run optimally
Use memory locking as part of real-time program initialization (Section 7.1.7)	Allows you to lock and unlock memory as needed	Reduces the memory available to processes and the UBC

The following sections describe these improvement guidelines in more detail.

7.1.1 Using the Latest Operating System Patches

Always install the latest operating system patches, which often contain performance enhancements.

Check the `/etc/motd` file to determine which patches you are running. Contact your customer service representative for information about installing patches.

7.1.2 Using the Latest Version of the Compiler

Always use the latest version of the compiler to build your application program. Usually, new versions include advanced optimizations.

Check the software on your system to ensure that you are using the latest version of the compiler.

7.1.3 Using Parallelism

To enhance parallelism, application developers working in Fortran or C should consider using the Kuch & Associates Preprocessor (KAP), which can have a significant impact on SMP performance. See the *Programmer's Guide* for details on KAP.

7.1.4 Optimizing Applications

Optimizing an application program can involve modifying the build process or modifying the source code. Various compiler and linker optimization levels can be used to generate more efficient user code. See the *Programmer's Guide* for more information on optimization.

Whether you are porting an application from a 32-bit system to Tru64 UNIX or developing a new application, never attempt to optimize an application until it has been thoroughly debugged and tested. If you are porting an application written in C, use the `lint` command with the `-Q` option or compile your program using the C compiler's `-check` option to identify possible portability problems that you may need to resolve.

7.1.5 Using Shared Libraries

Using shared libraries reduces the need for memory and disk space. When multiple programs are linked to a single shared library, the amount of physical memory used by each process can be significantly reduced.

However, shared libraries initially result in an execution time that is slower than if you had used static libraries.

7.1.6 Reducing Application Memory Requirements

You may be able to reduce an application's use of memory, which provides more memory resources for other processes or for file system caching. Follow these coding considerations to reduce your application's use of memory:

- Configure and tune applications according to the guidelines provided by the application's installation procedure. For example, you may be able to reduce an application's anonymous memory requirements, set parallel/concurrent processing attributes, size shared global areas and private caches, and set the maximum number of open/mapped files.
- You may want to use the `mmap` function instead of the `read` or `write` function in your applications. The `read` and `write` system calls require a page of buffer memory and a page of UBC memory, but `mmap` requires only one page of memory.
- Look for data cache collisions between heavily used data structures, which occur when the distance between two data structures allocated in memory is equal to the size of the primary (internal) data cache. If your data structures are small, you can avoid collisions by allocating them contiguously in memory. To do this, use a single `malloc` call instead of multiple calls.
- If an application uses large amounts of data for a short time, dynamically allocate the data with the `malloc` function instead of declaring it statically. When you have finished using dynamically allocated memory, it is freed for use by other data structures that occur later in the program. If you have limited memory resources, dynamically allocating data reduces an application's memory usage and can substantially improve performance.
- If an application uses the `malloc` function extensively, you may be able to improve its processing speed or decrease its memory utilization by using the function's control variables to tune memory allocation. See `malloc`(3) for details on tuning memory allocation.
- If your application fits in a 32-bit address space and allocates large amounts of dynamic memory by using structures that contain many pointers, you may be able to reduce memory usage by using the `-xtaso` option. The `-xtaso` option is supported by all versions of the C compiler (`-newc`, `-migrate`, and `-oldc` versions). To use the `-xtaso` option, modify your source code with a C-language pragma that controls pointer size allocations. See `cc`(1) for details.

See the *Programmer's Guide* for detailed information on process memory allocation.

7.1.7 Controlling Memory Locking

Real-time application developers should consider memory locking as a required part of program initialization. Many real-time applications remain locked for the duration of execution, but some may want to lock and unlock memory as the application runs. Memory-locking functions allow you to lock the entire process at the time of the function call and throughout the life of

the application. Locked pages of memory cannot be used for paging and the process cannot be swapped out.

Memory locking applies to a process's address space. Only the pages mapped into a process's address space can be locked into memory. When the process exits, pages are removed from the address space and the locks are removed.

Use the `mlockall` function to lock all of a process' address space. Locked memory remains locked until either the process exits or the application calls the `munlockall` function. Use the `ps` command to determine if a process is locked into memory and cannot be swapped out. See Section 12.3.2.

Memory locks are not inherited across a fork, and all memory locks associated with a process are unlocked on a call to the `exec` function or when the process terminates. See the *Guide to Realtime Programming* and `mlockall`(3) for more information.

Part 3

Tuning by Component

8

Managing System Resource Allocation

The Tru64 UNIX operating system sets resource limits at boot time. These limits control the size of system tables, virtual address space, and other system resources.

The default system resource limits are appropriate for most configurations. However, if your system has a large amount of memory, is running a program that requires extensive resources, or running a large-memory application, you may need to increase the system limits by modifying subsystem attributes.

This chapter describes system resource allocation and how to increase the following systemwide limits:

- Process limits (Section 8.1)
- Program size limits (Section 8.2)
- Address space limits (Section 8.3)
- Interprocess communication (IPC) limits (Section 8.4)
- Open file limits (Section 8.5)
- Aurema ARMTrech Suite (Section 8.6)

Instead of modifying systemwide limits, you can use the `setrlimit` function to control the consumption of system resources by a specific process and its child processes. See `setrlimit`(2) for more information.

8.1 Tuning Process Limits

Tru64 UNIX uses process limits that are appropriate for most configurations. However, if your applications are memory-intensive or you have a very-large memory (VLM) system or an Internet server (including, Web, proxy, firewall, or gateway servers), you may want to increase the process limits. Because increasing process limits increases the amount of wired memory in the system, increase the limits only if your system requires more resources.

The following sections describe how to increase these limits:

- System tables and data structures (Section 8.1.1)
- Maximum number of processes (Section 8.1.2)
- Maximum number of threads (Section 8.1.3)

For more information about the proc subystem attributes, see sys_attrs_proc(5).

8.1.1 Increasing System Tables and Data Structures

System algorithms use the proc subsystem attribute maxusers to size various system data structures and system tables, such as the system process table, which determines how many active processes can be running at one time.

Note

The value of the maxusers attribute is used to set the default values for some subsystem attributes that set system limits, including the max_proc_per_user, max_threads_per_user, min_free_vnodes, and name_cache_hash_size attributes.

Performance Benefit and Tradeoff

Increasing the value of maxusers provides more system resources to processes. However, increasing the resources available to users will increase the amount of wired memory.

You can modify the maxusers attribute without rebooting the system.

When to Tune

If you have a large-memory system or Internet server, or your system experiences a lack of resources, increase the value of the maxusers attribute. A lack of resources can be indicated by a No more processes, Out of processes, or pid table is full message.

Recommended Values

The default value assigned to the maxusers attribute depends on the amount of memory in the system. Table 8–1 shows the default value of the maxusers attribute for systems with various amounts of memory.

Table 8–1: Default Values for the maxusers Attribute

Size of Memory	Value of maxusers
Up to 256 MB	128
257 MB to 512 MB	256
513 MB to 1024 MB	512
1025 MB to 2048 MB	1024

Table 8–1: Default Values for the maxusers Attribute (cont.)

Size of Memory	Value of maxusers
2049 MB to 4096 MB	2048
4097 MB or more	2048

To determine an appropriate value for the maxusers attribute, double the default value until you notice a performance improvement. If you have an Internet server, you can increase the value of the maxusers attribute to 2048. We recommend that you not increase the value of the maxusers attribute to more than 2048.

If you increase the value of maxusers, you may want to increase the value of the max_vnodes attribute proportionally (see Section 8.5.1).

You must not decrease the default value of the maxusers attribute.

See Chapter 3 for information about modifying kernel subsystem attributes.

8.1.2 Increasing the Maximum Number of Processes

The proc subsystem attribute max_proc_per_user specifies the maximum number of processes that can be allocated at any one time to each user, except superuser.

Performance Benefit and Tradeoff

Increasing the value of max_proc_per_user provides more system resources to processes.

When to Tune

If your system experiences a lack of processes or you have a very-large memory (VLM) system or an Internet server, you may want to increase the value of the max_proc_per_user attribute.

You cannot modify the max_proc_per_user attribute without rebooting the system.

Recommended Values

The default value of the max_proc_per_user attribute is based on the maxusers attribute. If you want to increase the maximum number of processes, you can increase the value of the maxusers attribute (see Section 8.1.1). As an alternative, you can specify a value for the max_proc_per_user attribute that is equal to or greater than the maximum number of processes that will be running on the system at one time. If you have a Web server, these processes include CGI processes.

If you have an Internet server, increase the value of the max_proc_per_user attribute to 512.

If you specify a value of 0 for the max_proc_per_user attribute, there is no limit on processes.

See Chapter 3 for information about modifying kernel subsystem attributes.

8.1.3 Increasing the Maximum Number of Threads

The proc subsystem attribute max_threads_per_user specifies the maximum number of threads that can be allocated at any one time to each user, except superuser.

Performance Benefit and Tradeoff

Increasing the value of max_threads_per_user provides more system resources to processes.

You cannot modify the max_proc_per_user attribute without rebooting the system.

When to Tune

If your system experiences a lack of threads or you have a VLM system or an Internet server, you may want to increase the value of the max_threads_per_user attribute.

Recommended Values

The default value of the max_threads_per_user attribute is based on the value of the maxusers attribute. If you want to increase the maximum number of threads, you can modify the maxusers attribute (see Section 8.1.1). As an alternative, you can specify a value for the max_threads_per_user attribute that is equal to or greater than the maximum number of threads that are allocated at one time on the system. For example, you could increase the value of the max_threads_per_user attribute to 512.

On a very busy server with enough memory or an Internet server, increase the value of the max_threads_per_user attribute to 4096.

Setting the value of the max_threads_per_user attribute to 0 will remove the limit on threads.

If you specify a value of 0 for the max_threads_per_user attribute, there is no limit on threads.

See Chapter 3 for information about modifying kernel subsystem attributes.

8.2 Tuning Program Size Limits

If you are running a very large application, you may need to increase the values of the `proc` subsystem attributes that control program size limits. Some large programs and large-memory processes may not run unless you modify the default values of these attributes.

The following sections describe how to:

- Increase the maximum size of a user process stack (Section 8.2.1)
- Increase the maximum size of a user process data segment (Section 8.2.2)

For more information about the `proc` subystem attributes, see `sys_attrs_proc`(5).

8.2.1 Increasing the Size of a User Process Stack

The `proc` subsystem attributes `per_proc_stack_size` and `max_per_proc_stack_size` specify the default and maximum sizes of a user process stack. Some large programs and large-memory processes may not run unless you increase the default value of these attributes.

Performance Benefit and Tradeoff

Increasing the default and maximum sizes of a user process stack enables very large applications to run.

You cannot modify the `per_proc_stack_size` `max_per_proc_stack_size` attributes without rebooting the system.

When to Tune

If you are running a large program or a large-memory process, or if you receive `Cannot grow stack` messages, increase the default and maximum sizes of a user process stack.

Recommended Values

The default value of the `per_proc_stack_size` attribute is 8,388,608 bytes. The default value of the `max_per_proc_stack_size` attribute is 33,554,432 bytes. Choose values that are significantly less than the address space limit. See Section 8.3 for more information.

See Chapter 3 for information about modifying kernel subsystem attributes.

8.2.2 Increasing the Size of a User Process Data Segment

The `proc` subsystem attributes `per_proc_data_size` and `max_per_proc_data_size` specify the default and maximum sizes of a user

process data segment. Some large programs and large-memory processes may not run unless you increase the default values of these attributes.

Performance Benefit and Tradeoff

Increasing the default and maximum sizes of a user process data segment enables very large applications to run.

You cannot modify the `per_proc_data_size` and `max_per_proc_data_size` attributes without rebooting the system.

When to Tune

You may need to increase the values of the `per_proc_data_size` and `max_per_proc_data_size` attributes if you are running a large program or a large-memory process, if you receive an `Out of process memory` message, or the system is an Internet server.

Recommended Values

The default value of the `per_proc_data_size` is 134,217,728 bytes. The default value of the `max_per_proc_data_size` is 1 GB (1,073,741,824 bytes). Choose values that are significantly less than the address space limit. See Section 8.3 for information.

If you have an Internet server, increase the value of the `max_per_proc_data_size` attribute to 10 GB (10,737,418,240 bytes).

See Chapter 3 for information about modifying kernel subsystem attributes.

8.3 Tuning Address Space Limits

The `proc` subsystem attributes `per_proc_address_space` and `max_per_proc_address_space` specify the default and maximum amount of user process address space (number of valid virtual regions).

Performance Benefit and Tradeoff

Increasing the address space limit enables large programs to run, and improves the performance of memory-intensive applications. However, this causes a small increase in the demand for memory.

You cannot modify the `per_proc_address_space` and `max_per_proc_address_space` attributes without rebooting the system.

When to Tune

You may want to increase the address space limit if you are running a memory-intensive process, or if the system is an Internet server.

Recommended Values

The default value for the `per_proc_address_space` and `max_per_proc_address_space` attributes is 4 GB (4,294,967,296 bytes).

If you have an Internet server, increase the value of the `max_per_proc_address_space` attribute to 10 GB (10,737,418,240 bytes).

See Chapter 3 for information about modifying kernel attributes.

8.4 Tuning Interprocess Communication Limits

Interprocess communication (IPC) is the exchange of information between two or more processes. Some examples of IPC include messages, shared memory, semaphores, pipes, signals, process tracing, and processes communicating with other processes over a network.

The following sections describe how to:

- Increase the maximum size of a System V message (Section 8.4.1)
- Increase the maximum size of a System V message queue (Section 8.4.2)
- Increase the maximum size of messages on a System V queue (Section 8.4.3)
- Increase the maximum size of a system V shared memory region (Section 8.4.4)
- Increase the maximum number of shared memory regions attached to a process (Section 8.4.5)
- Modify shared page table sharing (Section 8.4.6)

The Tru64 UNIX operating system provides the following facilities for interprocess communication:

- Pipes — See the *Guide to Realtime Programming* for information about pipes.
- Signals — See the *Guide to Realtime Programming* for information.
- Sockets — See the *Network Programmer's Guide* for information.
- Streams — See the *Programmer's Guide: STREAMS* for information.
- X/Open Transport Interface (XTI) — See the *Network Programmer's Guide* for information.

If you are running processes that are memory-intensive, you may want to increase the values of some `ipc` subsystem attributes.

Table 8–2 describes the guidelines for increasing IPC limits and lists the performance benefits as well as tradeoffs.

Table 8–2: IPC Limits Tuning Guidelines

Guidelines	Performance Benefit	Tradeoff
Increase the maximum size of a System V message (Section 8.4.1)	May improve the performance of applications that can benefit from a large System V message size	Consumes a small amount of memory
Increase the maximum number of bytes on a System V message queue (Section 8.4.2)	May improve the performance of applications that can benefit from a large System V message queue	Consumes a small amount memory
Increase the maximum number of outstanding messages on a System V queue (Section 8.4.3)	May improve the performance of applications that benefit from having a large number of outstanding messages	Consumes a small amount of memory
Increase the maximum size of a System V shared memory region (Section 8.4.4)	May improve the performance of memory-intensive applications that can benefit from a large System V shared memory region	Consumes memory
Increase the maximum number of shared memory regions that can be attached to a process (Section 8.4.5)	May improve the performance of applications that attach many shared memory regions	May consume memory
Modify the shared page table limit (Section 8.4.6)	Enables memory-intensive or VLM systems to run efficiently	May consume memory

The following sections describe how to tune some System V attributes. See `sys_attrs_ipc`(5) for information about additional IPC subsystem attributes.

8.4.1 Increasing the Maximum Size of a System V Message

The `ipc` subsystem attribute `msg_max` specifies the maximum size of a System V message that an application can receive.

Performance Benefit and Tradeoff

Increasing the value of the `msg_max` attribute, may improve the performance of applications that can benefit from a System V message size that is larger than the default value. However, increasing this value will consume memory.

You cannot modify the msg_max attribute without rebooting the system.

When to Tune

If your applications can benefit from setting the default maximum size of a System V message to a value that is larger than 8192 bytes, you may want to increase the value of the msg_max attribute.

Recommended Value

The default value of the msg_max attribute is 8192 bytes (1 page).

See Chapter 3 for information about modifying kernel subsystem attributes.

8.4.2 Increasing the Maximum Size of a System V Message Queue

The ipc subsystem attribute msg_mnb, specifies the maximum number of bytes that can be in a System V message queue at one time.

A process cannot send a message to a queue if the number of bytes in the queue is greater than the limit specified by the msg_mnb attribute. When the limit is reached, the process sleeps and waits for this condition to be resolved.

Performance Benefit and Tradeoff

Increasing the value of the msg_mnb attribute may improve performance for applications that can benefit from a System V message queue that is larger than the default size. However, increasing this value will consume memory.

You cannot modify the msg_mnb attribute without rebooting the system.

When to Tune

You can track the use of IPC facilities with the ipcs -a command (see ipcs(1)). By looking at the current number of bytes and message headers in the queues, you can then determine whether you need to tune the System V message queue to diminish waiting.

Recommended Value

The default value of the msg_mnb attribute is 16,384 bytes.

See Chapter 3 for information about modifying kernel subsystem attributes.

8.4.3 Increasing the Maximum Number of Messages on a System V Queue

The ipc subsystem attribute msg_tql specifies the maximum number of messages that can be on a System V message queue; that is, the total number of messages that can be outstanding in the system.

Performance Benefit and Tradeoff

Increasing the value of the msg_tql attribute may improve the performance of applications that benefit from increasing the number of outstanding messages to a value that is larger than the default value. However, increasing the value of this attribute will consume memory.

You cannot modify the msg_tql attribute without rebooting the system.

When to Tune

You may want to increase the value of the msg_tql attribute if your applications can benefit from increasing the maximum number of outstanding messages to a value than is larger than 40.

You can track the use of IPC facilities with the ipcs -a command (see ipcs(1)). By looking at the current number of bytes and message headers in the queues, you can then determine whether you need to tune the System V message queue to diminish waiting.

Recommended Values

The default value of the msg_tql is 40.

See Chapter 3 for information about modifying kernel subsystem attributes.

8.4.4 Increasing the Maximum Size of a System V Shared Memory Region

The ipc subsystem attribute shm_max specifies the maximum size of a single System V shared memory region.

Performance Benefit and Tradeoff

Increasing the value of the shm_max attribute may improve the performance of memory-intensive applications that can benefit from a large System V shared memory region. However, increasing the value of the shm_max attribute will increase the demand for memory.

You cannot modify the shm_max attribute without rebooting the system.

When to Tune

If your applications are memory-intensive and can benefit from a System V shared memory region that is larger than the default value of 512 pages, you may want to increase the value of the shm_max attribute.

Recommended Value

The default value of the shm_max attribute is 4,194,304 bytes (512 pages).

See Chapter 3 for information about modifying kernel subsystem attributes.

8.4.5 Increasing the Maximum Number of Shared Memory Regions Attached to a Process

The ipc subsystem attribute shm_seg specifies the maximum number of System V shared memory regions that can be attached to a single process at any point in time.

As a design consideration, consider whether you will get better performance by using threads instead of shared memory.

Performance Benefit and Tradeoff

Increasing the number of System V shared memory regions that can be attached to a single process may improve the performance of applications that attach many shared memory regions.

Increasing the value of the shm_seg attribute will consume memory if the process attaches many shared memory regions.

You cannot modify the shm_seg attribute without rebooting the system.

When to Tune

You may want to increase the value of the shm_seg attribute if a process' attempt to attach a shared memory region exceeds the limit (the shmat function returns an EMFILE error).

Recommended Value

The default value of the shm_seg is 32.

See Chapter 3 for information about modifying kernel subsystem attributes.

8.4.6 Modifying Shared Page Table Sharing

Third-level page table sharing occurs when the size of a System V shared memory segment, as created by the shmget function, is equal to or larger than the value of the ipc subsystem attribute ssm_threshold.

Performance Benefit and Tradeoff

Increasing the shared page table limit restricts shared page tables to applications that create shared memory segments larger than 8 MB. However, this will increase the demand for memory.

You can disable page table sharing, if your applications cannot use shared page tables.

You can modify the ssm_threshold attribute without rebooting the system.

When to Tune

If you want to restrict page table sharing to applications that create shared memory segments larger than 8 MB, increase the value of the `ssm_threshold` attribute.

If your applications cannot use shared pages tables because of alignment restrictions, you may want to disable the sharing of page tables.

Recommended Value

The default value of the `ssm_threshold` attribute is 8 MB (8,388,608 bytes).

Setting the `ssm_threshold` attribute to 0 will disable the use of segmented shared memory.

See Chapter 3 for information about modifying kernel subsystem attributes.

8.5 Tuning the Open File Limits

The following sections describe how to:

- Increase the maximum number of open files (Section 8.5.1)
- Increase the maximum number of open file descriptors (Section 8.5.2)

For more information about the `proc` subystem attributes, see `sys_attrs_proc`(5).

8.5.1 Increasing the Maximum Number of Open Files

The kernel data structure for an open file is called a **vnode**. These are used by all file systems. The number of vnodes determines the number of open files. The allocation and deallocation of vnodes is dynamically handled by the operating system.

The `vfs` subsystem attribute `max_vnodes` specifies the size of the vnode cache, which is always equal to or more than the maximum number of open files in the system. You may need to increase the default value of this attribute to increase the maximum number of open files. Note that you can also accomplish this task by increasing the value of the `proc` subsystem attribute `maxusers`. See Section 8.1 for more information.

Performance Benefit and Tradeoff

Increasing the size of the vnode cache can improve the performance of applications that require many open files, but it will also consume memory.

You can modify the `max_vnodes` attribute without rebooting the system.

When to Tune

If your applications require many open files or you receive a message indicating you are out of vnodes, increase the default value of the max_vnodes attribute.

Recommended Value

The default value of the max_vnodes attribute is 5 percent of memory.

See Chapter 3 for information about modifying kernel subsystem attributes.

8.5.2 Increasing the Maximum Number of Open File Descriptors

You may want to increase the maximum number of open file descriptors for all processes or for a specific application. The proc subsystem attributes open_max_soft and open_max_hard control the maximum systemwide number of open file descriptors for each process.

The open file descriptor limits prevent runaway allocations, such as allocations within a loop that cannot be exited because of an error condition, from consuming all of the available file descriptors. If a process reaches the open_max_soft limit, a warning message is issued. If a process reaches the open_max_hard limit, the process is stopped.

Performance Benefit and Tradeoff

Improves the performance of applications that open many files.

You cannot modify the open_max_soft and open_max_hard attributes without rebooting the system.

When to Tune

If you have an application that requires many open files, you can increase the open file descriptor limit by increasing the values of the open_max_soft and open_max_hard attributes. However, increasing the open file descriptor limit may cause runaway allocations.

Recommended Values

The default value of the open_max_soft and open_max_hard attributes is 4,096, which is the maximum systemwide value that you can set in the /etc/sysconfigtab file.

If you have an application that requires many open files, you can increase the open file descriptor limit only for that application, instead of increasing the systemwide limit. To enable extended (64 KB) file descriptors for a specific application, follow these steps:

1. Set the setsysinfo system call's SSI_FD_NEWMAX operation parameter to 1, which sets the utask bit, enables up to 65,536 (64 KB) open file descriptors, and raises the process's hard file limit to 64 KB. This

setting is inherited by any child process. See setsysinfo(2) for more information.

2. Set the process's file descriptor soft limit to a value that is more than 4,096 (the default value) by using the setrlimit function as shown in the following code fragment:

```
# include <sys/resource.h>
struct rlimit *rlp;

rlp->rlim_cur = 6000;
rlp->rlim_max = 6000;
setrlimit(RLIMIT_NOFILE, rlp);
```

   This setting is inherited by any child process. See setrlimit(2) for more information.

3. This step is required only for applications that use the select function's fd_set parameter, which points to an I/O descriptor set (and a FD_CLR, FD_ISSET, FD_SET, or FD_ZERO macro) and can modify an I/O descriptor set. If you meet these qualifications, you can use one of two procedures, one that enables a static definition of the maximum number of file descriptors or one that enables a dynamic definition:

   - Static definition:

     Override the default value of 4,096 for FD_SETSIZE in the <sys/select.h> header file by specifying the maximum value of 65,536. You must specify this value before you include the <sys/time.h> header file (which also includes the <sys/select.h> header file) in the code, as follows:

```
# define FD_SETSIZE 65536
# include <sys/time.h>
```

     This setting is not inherited by child processes; therefore, FD_SETSIZE must be explicitly set in the code for each child process that requires 64-KB file descriptors.

   - Dynamic definition:

     Instead of using statically defined fd_set structures, you can use fd_set pointers in conjunction with a malloc function, which provides forward compatibility with any future changes to the maximum file descriptor limit. For example:

```
fd_set *fdp;

fdp = (fd_set *) malloc(
(fds_howmany(max_fds,FD_NFDBITS))*sizeof(fd_mask));
```

     The value for max_fds is the number of file descriptors to be manipulated. We recommend that you use the file descriptor soft limit for this value. All other keywords are defined in the

<sys/select.h> header file. The following code segment shows this choice:

```
# include <sys/time.h>
# include <sys/resource.h>

my_program()
{
fd_set *fdp;
struct rlimit rlim;
int max_fds;

getrlimit(RLIMIT_NOFILE, &rlim;);
max_fds = rlim.rlim_cur;

fdp = (fd_set *) malloc(
(fds_howmany(max_fds,FD_NFDBITS))*sizeof(fd_mask));

FD_SET(2, fdp);

for (;;) {
switch(select(max_fds, (fd_set *)0, fdp, (fd_set
*)0,
struct timeval *)0)) {
...
}
```

In addition, the vfs subsystem attribute max_vnodes must be set high enough for the needs of any application that requires a high number of descriptors. The max_vnodes attribute specifies the size of the vnode cache, and is set to 5 percent of system memory by default. See Section 8.5.1 for more information.

To disable support for up to 64-KB file descriptors for an application, set the setsysinfo system call's SSI_FD_NEWMAX operation parameter to 0, which disables the utask bit and returns the hard file limit to the default maximum of 4,096 open file descriptors. However, if the process is using more than 4,096 file descriptors, the setsysinfo system call will return an EINVAL error. In addition, if a calling process's hard or soft limit exceeds 4,096, the limit is set to 4 KB after the call is successful. This setting is inherited by any child process.

8.6 Aurema ARMTech Suite

Tru64 UNIX supports Aurema's ARMTech (Active Resource Management Technology) software product suite. This software provides enhanced resource management capabilities. The ARMTech suite provides the Tru64 UNIX administrator with the means to manage many operating system entities, such as Web sites or applications, in addition to users and groups.

See armtech(5) and the Aurema Web site:

http://www.aurema.com

9

Managing Disk Storage Performance

There are various ways that you can manage your disk storage. Depending on your performance and availability needs, you can use static disk partitions, the Logical Storage Manager (LSM), hardware RAID, or a combination of these solutions.

The disk storage configuration can have a significant impact on system performance, because disk I/O is used for file system operations and also by the virtual memory subsystem for paging and swapping.

You may be able to improve disk I/O performance by following the configuration and tuning guidelines described in this chapter, which describes the following:

- Improving overall disk I/O performance by distributing the I/O load (Section 9.1)
- Monitoring the distribution of disk I/O (Section 9.2)
- Managing LSM performance (Section 9.3)
- Managing hardware RAID subsystem performance (Section 9.4)
- Managing Common Access Method (CAM) performance (Section 9.5)

Not all guidelines are appropriate for all disk storage configurations. Before applying any guideline, be sure that you understand your workload resource model, as described in Section 1.8, and the guideline's benefits and tradeoffs.

9.1 Guidelines for Distributing the Disk I/O Load

Distributing the disk I/O load across devices helps to prevent a single disk, controller, or bus from becoming a bottleneck. It also enables simultaneous I/O operations.

For example, if you have 16 GB of disk storage, you may get better performance from sixteen 1-GB disks rather than four 4-GB disks, because using more spindles (disks) may allow more simultaneous operations. For random I/O operations, 16 disks may be simultaneously seeking instead of four disks. For large sequential data transfers, 16 data streams can be simultaneously working instead of four data streams.

Use the following guidelines to distribute the disk I/O load:

- Stripe data or disks.

  RAID0 (data or disk striping) enables you to efficiently distribute data across the disks. See Section 11.2.1.5 for detailed information about the benefits of striping. Note that availability decreases as you increase the number of disks in a striped array.

  To stripe data, use LSM (see Section 9.3). To stripe disks, use a hardware RAID subsystem (see Section 9.4).

  As an alternative to data or disk striping, you can use the Advanced File System (AdvFS) to stripe individual files across disks in a file domain. However, do not stripe a file and also the disk on which it resides. See Section 11.2 for more information.

- Use RAID5.

  RAID5 distributes disk data and parity data across disks in an array to provide high data availability and to improve read performance. However, RAID5 decreases write performance in a nonfailure state, and decreases read and write performance in a failure state. RAID5 can be used for configurations that are mainly read-intensive. As a cost-efficient alternative to mirroring, you can use RAID5 to improve the availability of rarely accessed data.

  To create a RAID5 configuration, use LSM (see Section 9.3) or a hardware RAID subsystem (Section 9.4).

- Distribute frequently used file systems across disks and, if possible, different buses and controllers.

  Place frequently used file systems on different disks and, if possible, different buses and controllers. Directories containing executable files or temporary files, such as `/var`, `/usr`, and `/tmp`, are often frequently accessed. If possible, place `/usr` and `/tmp` on different disks.

  You can use the AdvFS `balance` command to balance the percentage of used space among the disks in an AdvFS file domain. See Section 11.2.1.4 for information.

- Distribute swap I/O across devices.

  To make paging and swapping more efficient and help prevent any single adapter, bus, or disk from becoming a bottleneck, distribute swap space across multiple disks. Do not put multiple swap partitions on the same disk.

  You can also use LSM to mirror your swap space. See Section 9.3 for more information.

  See Section 12.2 for more information about configuring swap devices for high performance.

Section 9.2 describes how to monitor the distribution of disk I/O.

9.2 Monitoring the Distribution of Disk I/O

Table 9–1 describes some commands that you can use to determine if your disk I/O is being distributed.

Table 9–1: Disk I/O Distribution Monitoring Tools

Tool	Description	Reference
`showfdmn`	Displays information about AdvFS file domains and determines if files are evenly distributed across AdvFS volumes.	Section 11.2.2.3
`advfsstat`	Displays information about AdvFS file domain and fileset usage, and provides performance statistics information for AdvFS file domains and filesets that you can use to determine if the file system I/O is evenly distributed.	Section 11.2.2.1
`swapon`	Displays the swap space configuration and usage. It displays the total amount of allocated swap space, the amount of swap space that is being used, and the amount of free swap space.	Section 12.3.3
`volstat`	Displays performance statistics for LSM objects and provides information about LSM volume and disk usage that you can use to characterize and understand your I/O workload, including the read/write ratio, the average transfer size, and whether disk I/O is evenly distributed.	Section 9.3 or the *Logical Storage Manager* documentation.
`iostat`	Displays disk I/O statistics and provides information about which disks are being used the most.	Section 9.2.1

9.2.1 Displaying Disk Usage by Using the iostat Command

For the best performance, disk I/O should be evenly distributed across disks. Use the `iostat` command to determine which disks are being used the most. The command displays disk I/O statistics for disks, in addition to terminal and CPU statistics.

An example of the `iostat` command is as follows; output is provided in one-second intervals:

```
# /usr/ucb/iostat 1
     tty     floppy0     dsk0      dsk1    cdrom0      cpu
 tin tout   bps tps  bps tps  bps tps  bps tps  us ni sy id
   1   73     0   0   23   2   37   3    0   0   5  0 17 79
   0   58     0   0   47   5  204  25    0   0   8  0 14 77
   0   58     0   0    8   1   62   1    0   0  27  0 27 46
```

The `iostat` command output displays the following information:

- The first line of the `iostat` command output is the average since boot time, and each subsequent report is for the last interval.
- For each disk (`dsk`*n*), the number of KB transferred per second (`bps`) and the number of transfers per second (`tps`).
- For the system (`cpu`), the percentage of time the CPU has spent in user state running processes either at their default priority or preferred priority (`us`), in user mode running processes at a less favored priority (`ni`), in system mode (`sy`), and in idle mode (`id`). This information enables you to determine how disk I/O is affecting the CPU. User mode includes the time the CPU spent executing library routines. System mode includes the time the CPU spent executing system calls.

The `iostat` command can help you to do the following:

- Determine which disk is being used the most and which is being used the least. This information will help you determine how to distribute your file systems and swap space. Use the `swapon -s` command to determine which disks are used for swap space.
- Determine if the system is disk bound. If the `iostat` command output shows a lot of disk activity and a high system idle time, the system may be disk bound. You may need to balance the disk I/O load, defragment disks, or upgrade your hardware.
- Determine if an application is written efficiently. If a disk is doing a large number of transfers (the `tps` field) but reading and writing only small amounts of data (the `bps` field), examine how your applications are doing disk I/O. The application may be performing a large number of I/O operations to handle only a small amount of data. You may want to rewrite the application if this behavior is not necessary.

9.3 Managing Storage with LSM

The Logical Storage Manager (LSM) provides flexible storage management, improved disk I/O performance, and high data availability, with little additional overhead. Although any type of system can benefit from LSM, it is especially suited for configurations with large numbers of disks or configurations that regularly add storage.

LSM allows you to set up unique pools of storage that consist of multiple disks. From these disk groups, you can create virtual disks (LSM volumes), which are used in the same way as disk partitions. You can create UFS or AdvFS file systems on a volume, use a volume as a raw device, or create volumes on top of RAID storage sets.

Because there is no direct correlation between an LSM volume and a physical disk, file system or raw I/O can span disks. You can easily add disks to and remove disks from a disk group, balance the I/O load, and perform other storage management tasks.

In addition, LSM provides high performance and high availability by using RAID technology. LSM is often referred to as **software RAID**. LSM configurations can be more cost-effective and less complex than a hardware RAID subsystem. Note that LSM RAID features require a license.

9.3.1 LSM Features

LSM provides the following basic disk management features that do not require a license:

- Disk concatenation enables you to create a large volume from multiple disks.
- Load balancing transparently distributes data across disks.
- Configuration database load-balancing automatically maintains an optimal number of LSM configuration databases in appropriate locations without manual intervention.
- The `volstat` command provides detailed LSM performance information.

The following LSM features require a license:

- RAID0 (striping) distributes data across disks in an array. Striping is useful if you quickly transfer large amounts of data, and also enables you to balance the I/O load from multi-user applications across multiple disks. LSM striping provides significant I/O performance benefits with little impact on the CPU.
- RAID1 (mirroring) maintains copies of data on different disks and reduces the chance that a single disk failure will cause the data to be unavailable.

- RAID5 (parity RAID) provides data availability through the use of parity and distributes data and parity across disks in an array.
- Mirrored root file system and swap space improves availability.
- Hot-spare support provides an automatic reaction to I/O failures on mirrored or RAID5 objects by relocating the affected objects to spare disks or other free space.
- Dirty-region logging (DRL) improves the recovery time of mirrored volumes after a system failure.
- A graphical user interface (GUI) enables easy disk management and provides detailed performance information.

To obtain the best LSM performance, follow the configuration and tuning guidelines described in the *Logical Storage Manager* manual.

9.4 Managing Hardware RAID Subsystem Performance

Hardware RAID subsystems provide RAID functionality for high performance and high availability, relieve the CPU of disk I/O overhead, and enable you to connect many disks to a single I/O bus or in some cases, multiple buses. There are various types of hardware RAID subsystems with different performance and availability features, but they all include a RAID controller, disks in enclosures, cabling, and disk management software.

RAID storage solutions range from low-cost backplane RAID array controllers to cluster-capable RAID array controllers that provide extensive performance and availability features, such as write-back caches and complete component redundancy.

Hardware RAID subsystems use disk management software, such as the RAID Configuration Utility (RCU) and the StorageWorks Command Console (SWCC) utility, to manage the RAID devices. Menu-driven interfaces allow you to select RAID levels.

Use hardware RAID to combine multiple disks into a single storage set that the system sees as a single unit. A storage set can consist of a simple set of disks, a striped set, a mirrored set, or a RAID set. You can create LSM volumes, AdvFS file domains, or UFS file systems on a storage set, or you can use the storage set as a raw device.

The following sections discuss the following RAID hardware topics:

- Hardware RAID features (Section 9.4.1)
- Hardware RAID products (Section 9.4.2)
- Guidelines for hardware RAID configurations (Section 9.4.3)

See the hardware RAID product documentation for detailed configuration information.

9.4.1 Hardware RAID Features

Hardware RAID storage solutions range from low-cost backplane RAID array controllers to cluster-capable RAID array controllers that provide extensive performance and availability features. All hardware RAID subsystems provide you with the following features:

- A RAID controller that relieves the CPU of the disk I/O overhead
- Increased disk storage capacity

  Hardware RAID subsystems allow you to connect a large number of disks to a single I/O bus or, in some cases, multiple buses. In a typical storage configuration, you attach a disk storage shelf to a system by using a SCSI bus connected to a host bus adapter installed in a I/O bus slot. However, you can connect a limited number of disks to a SCSI bus, and systems have a limited number of I/O bus slots.

  In contrast, hardware RAID subsystems contain multiple internal SCSI buses that can be connected to a system by using a single I/O bus slot.
- Read cache

  A read cache improves I/O read performance by holding data that it anticipates the host will request. If a system requests data that is already in the read cache (a cache hit), the data is immediately supplied without having to read the data from disk. Subsequent data modifications are written both to disk and to the read cache (write-through caching).
- Write-back cache

  Hardware RAID subsystems support write-back caches (as a standard or an optional feature), which can improve I/O write performance while maintaining data integrity. A write-back cache decreases the latency of many small writes, and can improve Internet server performance because writes appear to be written immediately. Applications that perform few writes will not benefit from a write-back cache.

  With write-back caching, data intended to be written to disk is temporarily stored in the cache, consolidated, and then periodically written (flushed) to disk for maximum efficiency. I/O latency is reduced by consolidating contiguous data blocks from multiple host writes into a single unit.

  A write-back cache must have an uninterruptible power source (UPS) to protect against data loss and corruption.
- RAID support

All hardware RAID subsystems support RAID0 (disk striping), RAID1 (disk mirroring), and RAID5. High-performance RAID array subsystems also support RAID3 and dynamic parity RAID. See Section 1.3.1 for information about RAID levels.

- Non-RAID disk array capability or "just a bunch of disks" (JBOD)
- Component hot swapping and hot sparing

  Hot-swap support allows you to replace a failed component while the system continues to operate. Hot-spare support allows you to automatically use previously installed components if a failure occurs.
- Graphical user interface (GUI) for easy management and monitoring

9.4.2 Hardware RAID Products

There are different types of hardware RAID subsystems, which provide various degrees of performance and availability at various costs. HP supports the following hardware RAID subsystems:

- Backplane RAID array storage subsystems

  These entry-level subsystems, such as those utilizing the RAID Array 230/Plus storage controller, provide a low-cost hardware RAID solution and are designed for small and midsize departments and workgroups.

  A backplane RAID array storage controller is installed in a PCI bus slot and acts as both a host bus adapter and a RAID controller.

  Backplane RAID array subsystems provide RAID functionality (0, 1, 0+1, and 5), an optional write-back cache, and hot-swap functionality.
- High-performance RAID array subsystems

  These subsystems, such as the RAID Array 450 subsystem, provide extensive performance and availability features and are designed for client/server, data center, and medium to large departmental environments.

  A high-performance RAID array controller, such as an HSZ80 controller, is connected to a system through an ultrawide differential SCSI bus and a high-performance host bus adapter installed in an I/O bus slot.

  High-performance RAID array subsystems provide RAID functionality (0, 1, 0+1, 3, 5, and dynamic parity RAID), dual-redundant controller support, scalability, storage set partitioning, a standard UPS write-back cache, and components that can be hotswapped.
- Enterprise Storage Arrays (ESA)/Modular storage array (MSA)

  These preconfigured high-performance hardware RAID subsystems, such as the RAID Array 12000, provide the highest performance, availability, and disk capacity of any RAID subsystem. They are used for high

transaction-oriented applications and high bandwidth decision-support applications.

ESAs support all major RAID levels, including dynamic parity RAID; fully redundant components that can be hot-swapped; a standard UPS write-back cache; and centralized storage management.

See the HP *Logical Storage Manager Version 5.1B QuickSpecs* for detailed information about hardware RAID subsystem features.

9.4.3 Hardware RAID Configuration Guidelines

Table 9–2 describes the hardware RAID subsystem configuration guidelines and lists performance benefits as well as tradeoffs.

Table 9–2: Hardware RAID Subsystem Configuration Guidelines

Guideline	Performance Benefit	Tradeoff
Evenly distribute disks in a storage set across different buses (Section 9.4.3.1)	Improves performance and helps to prevent bottlenecks	None
Use disks with the same data capacity in each storage set (Section 9.4.3.2)	Simplifies storage management	None
Use an appropriate stripe size (Section 9.4.3.3)	Improves performance	None
Mirror striped sets (Section 9.4.3.4)	Provides availability and distributes disk I/O performance	Increases configuration complexity and may decrease write performance
Use a write-back cache (Section 9.4.3.5)	Improves write performance, especially for RAID5 storage sets	Cost of hardware
Use dual-redundant RAID controllers (Section 9.4.3.6)	Improves performance, increases availability, and prevents I/O bus bottlenecks	Cost of hardware
Install spare disks (Section 9.4.3.7)	Improves availability	Cost of disks
Replace failed disks promptly (Section 9.4.3.7)	Improves performance	None

The following sections describe some of these guidelines. See your RAID subsystem documentation for detailed configuration information.

9.4.3.1 Distributing Storage Set Disks Across Buses

You can improve performance and help to prevent bottlenecks by distributing storage set disks evenly across different buses.

In addition, make sure that the first member of each mirrored set is on a different bus.

9.4.3.2 Using Disks with the Same Data Capacity

Use disks with the same capacity in a storage set. This simplifies storage management and reduces wasted disk space.

9.4.3.3 Choosing the Correct Hardware RAID Stripe Size

You must understand how your applications perform disk I/O before you can choose the stripe (chunk) size that will provide the best performance benefit. See Section 1.8 for information about identifying a resource model for your system.

Here are some guidelines for stripe sizes:

- If the stripe size is large compared to the average I/O size, each disk in a stripe set can respond to a separate data transfer. I/O operations can then be handled in parallel, which increases sequential write performance and throughput. This can improve performance for environments that perform large numbers of I/O operations, including transaction processing, office automation, and file services environments, and for environments that perform multiple random read and write operations.
- If the stripe size is smaller than the average I/O operation, multiple disks can simultaneously handle a single I/O operation, which can increase bandwidth and improve sequential file processing. This is beneficial for image processing and data collection environments. However, do not make the stripe size so small that it will degrade performance for large sequential data transfers.

For example, if you use an 8-KB stripe size, small data transfers will be distributed evenly across the member disks, but a 64-KB data transfer will be divided into at least 8 data transfers.

In addition, the following guidelines can help you choose the correct stripe size:

- Raw disk I/O operations

  If your applications are doing I/O to a raw device and not a file system, use a stripe size that distributes a single data transfer evenly across the member disks. For example, if the typical I/O size is 1 MB and you have a four-disk array, you could use a 256-KB stripe size. This would

distribute the data evenly among the four member disks, with each doing a single 256-KB data transfer in parallel.

- Small file system I/O operations

  For small file system I/O operations, use a stripe size that is a multiple of the typical I/O size (for example, four to five times the I/O size). This will help to ensure that the I/O is not split across disks.

- I/O to a specific range of blocks

  Choose a stripe size that will prevent any particular range of blocks from becoming a bottleneck. For example, if an application often uses a particular 8-KB block, you may want to use a stripe size that is slightly larger or smaller than 8 KB or is a multiple of 8 KB to force the data onto a different disk.

9.4.3.4 Mirroring Striped Sets

Striped disks improve I/O performance by distributing the disk I/O load. However, striping decreases availability because a single disk failure will cause the entire stripe set to be unavailable. To make a stripe set highly available, you can mirror the stripe set.

9.4.3.5 Using a Write-Back Cache

RAID subsystems support, either as a standard or an optional feature, a nonvolatile (battery-backed) write-back cache that can improve disk I/O performance while maintaining data integrity. A write-back cache improves performance for systems that perform large numbers of writes and for RAID5 storage sets. Applications that perform few writes will not benefit from a write-back cache.

With write-back caching, data intended to be written to disk is temporarily stored in the cache and then periodically written (flushed) to disk for maximum efficiency. I/O latency is reduced by consolidating contiguous data blocks from multiple host writes into a single unit.

A write-back cache improves performance, especially for Internet servers, because writes appear to be written immediately. If a failure occurs, upon recovery, the RAID controller detects any unwritten data that still exists in the write-back cache and writes the data to disk before enabling normal controller operations.

A write-back cache must be backed up with an uninterruptible power source (UPS) to protect against data loss and corruption.

If you are using an HSZ40, HSZ50, HSZ70, or HSZ80 RAID controller with a write-back cache, the following guidelines may improve performance:

- Set `CACHE_POLICY` to B.

- Set CACHE_FLUSH_TIMER to a minimum of 45 (seconds).
- Enable the write-back cache (WRITEBACK_CACHE) for each unit, and set the value of MAXIMUM_CACHED_TRANSFER_SIZE to a minimum of 256.

See the RAID subsystem documentation for more information about using the write-back cache.

9.4.3.6 Using Dual-Redundant Controllers

If supported by your RAID subsystem, you can use a dual-redundant controller configuration and balance the number of disks across the two controllers. This can improve performance, increase availability, and prevent I/O bus bottlenecks.

9.4.3.7 Using Spare Disks to Replace Failed Disks

Install predesignated spare disks on separate controller ports and storage shelves. This will help you to maintain data availability and recover quickly if a disk failure occurs.

9.5 Managing CAM Performance

The Common Access Method (CAM) is the operating system interface to the hardware. CAM maintains pools of buffers that are used to perform I/O. Each buffer takes approximately 1 KB of physical memory. Monitor these pools and tune them if necessary.

You may be able to modify the following io subsystem attributes to improve CAM performance:

- cam_ccb_pool_size — The initial size of the buffer pool free list at boot time. The default is 200.
- cam_ccb_low_water — The number of buffers in the pool free list at which more buffers are allocated from the kernel. CAM reserves this number of buffers to ensure that the kernel always has enough memory to shut down runaway processes. The default is 100.
- cam_ccb_increment — The number of buffers either added or removed from the buffer pool free list. Buffers are allocated on an as-needed basis to handle immediate demands, but are released in a more measured manner to guard against spikes. The default is 50.

If the I/O pattern associated with your system tends to have intermittent bursts of I/O operations (I/O spikes), increasing the values of the cam_ccb_pool_size and cam_ccb_increment attributes may improve performance.

You may be able to diagnose CAM performance problems by using dbx to examine the ccmn_bp_head data structure, which provides statistics on the buffer structure pool that is used for raw disk I/O. The information provided is the current size of the buffer structure pool (num_bp) and the wait count for buffers (bp_wait_cnt).

For example:

```
# /usr/ucb/dbx –k /vmunix /dev/mem
(dbx) print ccmn_bp_head
struct {
    num_bp = 50
    bp_list = 0xffffffff81f1be00
    bp_wait_cnt = 0
}
(dbx)
```

If the value for the bp_wait_cnt field is not 0, CAM has run out of buffer pool space. If this situation persists, you may be able to eliminate the problem by changing one or more of the CAM subsystem attributes described in this section.

10

Managing Network Performance

The network used to communicate between the client and server does not normally cause a performance problem. There are, however, two conditions to look out for: network delays and high retransmission rates. If the Ethernet is overutilized, clients experience long delays waiting for a free slot to send requests. An Ethernet utilization of over 50 percent often indicates excessive network delay.

Network topology often contributes to excessive delay. If clients are located across many gateways from the servers that they use often, their requests experience long delays. You may be able to solve the problem by restructuring the network topology to evenly distribute the load.

This chapter describes how to manage Tru64 UNIX network subsystem performance. The following sections describe how to:

- Monitor the network subsystem (Section 10.1)
- Tune the network subsystem (Section 10.2)

10.1 Gathering Network Information

Table 10–1 describes the commands you can use to obtain information about network operations.

Table 10–1: Network Monitoring Tools

Tool	Description	Reference
`netstat`	Displays network statistics and a list of active sockets for each protocol, information about network routes, and cumulative statistics for network interfaces, including the number of incoming and outgoing packets and packet collisions. Also displays information about memory used for network operations.	Section 2.4.5

Table 10–1: Network Monitoring Tools (cont.)

Tool	Description	Reference
`sobacklog_hiwat` attribute	Reports the maximum number of pending requests to any server socket, and allows you to display the maximum number of pending requests to any server socket in the system.	Section 10.1.1
`sobacklog_drops` attribute	Reports the number of backlog drops that exceed a socket backlog limit, and allows you to display the number of times the system dropped a received SYN packet because the number of queued `SYN_RCVD` connections for a socket equaled the socket backlog limit.	Section 10.1.1
`somaxconn_drops` attribute	Reports the number of drops that exceed the value of the `somaxconn` attribute, and allows you to display the number of times the system dropped a received SYN packet because the number of queued `SYN_RCVD` connections for a socket equaled the upper limit on the backlog length (`somaxconn` attribute).	Section 10.1.1
`ping`	Determines if a system can be reached on the network and sends an Internet Control Message Protocol (ICMP) echo request to a host to determine if a host is running and reachable, and to determine if an IP router is reachable. Enables you to isolate network problems, such as direct and indirect routing problems.	See `ping`(8)

Table 10–1: Network Monitoring Tools (cont.)

Tool	Description	Reference
`tcpdump`	Monitors network interface packets on a network interface. You can specify the interface on which to listen, the direction of the packet transfer, or the type of protocol traffic to display. The `tcpdump` command allows you to monitor the network traffic associated with a particular network service and to identify the source of a packet. It lets you determine whether requests are being received or acknowledged, or to determine the source of network requests, in the case of slow network performance. Your kernel must be configured with the `packetfilter` option to use the command.	See Section 2.4.4
`traceroute`	Displays the packet route to a network host and tracks the route network packets follow from gateway to gateway.	See `traceroute`(8)

10.1.1 Checking Socket Listen Queue Statistics by Using the sysconfig Command

You can determine whether you need to increase the socket listen queue limit by using the `sysconfig -q socket` command to display the values of the following attributes:

- `sobacklog_hiwat` — Allows you to monitor the maximum number of pending requests to any server socket in the system. The initial value is 0.
- `sobacklog_drops` — Allows you to monitor the number of times the system dropped a received SYN packet because the number of queued `SYN_RCVD` connections for a socket equaled the socket backlog limit. The initial value is 0.
- `somaxconn_drops` — Allows you to monitor the number of times the system dropped a received SYN packet because the number of queued `SYN_RCVD` connections for the socket equaled the upper limit on the backlog length (`somaxconn` attribute). The initial value is 0.

We recommend that the value of the sominconn attribute equal the value of the somaxconn attribute. If so, the value of somaxconn_drops will have the same value as sobacklog_drops.

However, if the value of the sominconn attribute is 0 (the default), and if one or more server applications uses an inadequate value for the backlog argument to its listen system call, the value of sobacklog_drops may increase at a rate that is faster than the rate at which the somaxconn_drops counter increases. If this occurs, you may want to increase the value of the sominconn attribute.

See Section 10.2.3 for information on tuning socket listen queue limits.

10.2 Tuning the Network Subsystem

Most resources used by the network subsystem are allocated and adjusted dynamically; however, there are some tuning guidelines that you can use to improve performance, particularly with systems that are Internet servers, including Web, proxy, firewall, and gateway servers.

Network performance is affected when the supply of resources is unable to keep up with the demand for resources. The following two conditions can cause this to occur:

- A problem with one or more hardware or software network components
- A workload (network traffic) that consistently exceeds the capacity of the available resources, although everything appears to be operating correctly

Neither of these problems are network tuning issues. In the case of a problem on the network, you must isolate and eliminate the problem. In the case of high network traffic (for example, the hit rate on a Web server has reached its maximum value while the system is 100 percent busy), you must either redesign the network and redistribute the load, reduce the number of network clients, or increase the number of systems handling the network load.

See the *Network Programmer's Guide* and the *Network Administration: Connections* manual for information on how to resolve network problems.

Table 10–2 lists network subsystem tuning guidelines and performance benefits as well as tradeoffs.

Table 10–2: Network Tuning Guidelines

Guidelines	Performance Benefit	Tradeoff
Increase the size of the hash table that the kernel uses to look up TCP control blocks (Section 10.2.1)	Improves the TCP control block lookup rate and increases the raw connection rate	Slightly increases the amount of wired memory
Increase the number of TCP hash tables (Section 10.2.2)	Reduces hash table lock contention for SMP systems	Slightly increases the amount of wired memory
Increase the limits for partial TCP connections on the socket listen queue (Section 10.2.3)	Improves throughput and response time on systems that handle a large number of connections	Consumes memory when pending connections are retained in the queue
Increase the number of outgoing connection ports (Section 10.2.4)	Allows more simultaneous outgoing connections	None
Modify the range of outgoing connection ports (Section 10.2.5)	Allows you to use ports from a specific range	None
Disable the use of PMTU discovery (Section 10.2.6)	Improves the efficiency of servers that handle remote traffic from many clients	May reduce server efficiency for LAN traffic
Increase the number of IP input queues (Section 10.2.7)	Reduces IP input queue lock contention for SMP systems	None
Enable `mbuf` cluster compression (Section 10.2.8)	Improves efficiency of network memory allocation	None
Enable TCP keepalive functionality (Section 10.2.9)	Enables inactive socket connections to time out	None
Increase the size of the kernel interface alias table (Section 10.2.10)	Improves the IP address lookup rate for systems that serve many domain names	Slightly increases the amount of wired memory
Make partial TCP connections time out more quickly (Section 10.2.11)	Prevents clients from overfilling the socket listen queue	A short time limit may cause viable connections to break prematurely
Make the TCP connection context time out more quickly at the end of the connection (Section 10.2.12)	Frees connection resources sooner	Reducing the timeout limit increases the potential for data corruption; use caution if you apply this guideline

Table 10–2: Network Tuning Guidelines (cont.)

Guidelines	Performance Benefit	Tradeoff
Reduce the TCP retransmission rate (Section 10.2.13)	Prevents premature retransmissions and decreases congestion	A long retransmit time is not appropriate for all configurations
Enable the immediate acknowledgment of TCP data (Section 10.2.14)	Can improve network performance for some connections	May adversely affect network bandwidth
Increase the TCP maximum segment size (Section 10.2.15)	Allows sending more data per packet	May result in fragmentation at the router boundary
Increase the size of the transmit and receive socket buffers (Section 10.2.3)	Buffers more TCP packets per socket	May decrease available memory when the buffer space is being used
Increase the size of the transmit and receive buffers for a UDP socket (Section 10.2.16)	Helps to prevent dropping UDP packets	May decrease available memory when the buffer space is being used
Allocate sufficient memory to the UBC (Section 11.1.3)	Improves disk I/O performance	May decrease the physical memory available to processes
Increase the maximum size of a socket buffer (Section 10.2.17)	Allows large socket buffer sizes	Consumes memory resources
Prevent dropped input packets (Section 10.2.18)	Allows high network loads	None

The following sections describe these tuning guidelines in more detail.

See Chapter 3 for information about modifying kernel subsystem attributes.

10.2.1 Improving the Lookup Rate for TCP Control Blocks

You can modify the size of the hash table that the kernel uses to look up Transmission Control Protocol (TCP) control blocks. The `inet` subsystem attribute `tcbhashsize` specifies the number of hash buckets in the kernel TCP connection table (the number of buckets in the `inpcb` hash table).

Performance Benefit and Tradeoff

The kernel must look up the connection block for every TCP packet it receives, so increasing the size of the table can speed the search and improve performance. This results in a small increase in wired memory.

You can modify the `tcbhashsize` attribute without rebooting the system.

When to Tune

Increase the number of hash buckets in the kernel TCP connection table if you have an Internet server.

Recommended Values

The default value of the `tcbhashsize` attribute is 512. For Internet servers, set the `tcbhashsize` attribute to 16384.

10.2.2 Increasing the Number of TCP Hash Tables

Because the kernel must look up the connection block for every Transmission Control Protocol (TCP) packet it receives, a bottleneck may occur at the TCP hash table in SMP systems. Increasing the number of tables distributes the load and may improve performance. The `inet` subsystem attribute `tcbhashnum` specifies the number of TCP hash tables.

Performance Benefit and Tradeoff

For SMP systems, you may be able to reduce hash table lock contention by increasing the number of hash tables that the kernel uses to look up TCP control blocks. This will slightly increase wired memory.

You cannot modify the `tcbhashnum` attribute without rebooting the system.

When to Tune

Increase the number of TCP hash tables if you have an SMP system that is an Internet server.

Recommended Values

The minimum and default values of the `tcbhashnum` attribute are 1; the maximum value is 64. For busy Internet server SMP systems, you can increase the value of the `tcbhashnum` attribute to 16. If you increase this attribute, you should also increase the size of the hash table. See Section 10.2.1 for information.

We recommend that you make the value of the `tcbhashnum` attribute the same as the value of the `inet` subsystem attribute `ipqs`. See Section 10.2.7 for information.

10.2.3 Tuning the TCP Socket Listen Queue Limits

You may be able to improve performance by increasing the limits for the socket listen queue (only for TCP). The `socket` subsystem attribute `somaxconn` specifies the maximum number of pending TCP connections (the socket listen queue limit) for each server socket. If the listen queue connection limit is too small, incoming connect requests may be dropped.

Note that pending TCP connections can be caused by lost packets in the Internet or denial of service attacks.

The socket subsystem attribute sominconn specifies the minimum number of pending TCP connections (backlog) for each server socket. The attribute controls how many SYN packets can be handled simultaneously before additional requests are discarded. The value of the sominconn attribute overrides the application-specific backlog value, which may be set too low for some server software.

Performance Benefit and Tradeoff

To improve throughput and response time with fewer drops, you can increase the value of the somaxconn attribute.

If you want to improve performance without recompiling an application or if you have an Internet server, increase the value of the sominconn attribute. Increasing the value of this attribute can also prevent a client from saturating a socket listen queue with erroneous TCP SYN packets.

You can modify the somaxconn and sominconn attributes without rebooting the system. However, sockets that are already open will continue to use the previous socket limits until the applications are restarted.

When to Tune

Increase the socket listen queue limits if you have an Internet server or a busy system that has many pending connections and is running applications generating a large number of connections.

Monitor the sobacklog_hiwat, sobacklog_drops, and somaxconn_drops attributes to determine if socket queues are overflowing. If so, you may need to increase the socket listen queue limits. See Section 10.1.1 for information.

Recommended Values

The default value of the somaxconn attribute is 1024. For Internet servers, set the value of the somaxconn attribute to the maximum value of 65535.

The default value of the sominconn attribute is 0. To improve performance without recompiling an application and for Internet servers, set the value of the sominconn attribute to the maximum value of 65535.

If a client is saturating a socket listen queue with erroneous TCP SYN packets, effectively blocking other users from the queue, increase the value of the sominconn attribute to 65535. If the system continues to drop incoming SYN packets, you can decrease the value of the inet subsystem attribute tcp_keepinit to 30 (15 seconds).

The value of the sominconn attribute should be the same as the value of the somaxconn attribute.

10.2.4 Increasing the Number of Outgoing Connection Ports

When a TCP or UDP application creates an outgoing connection, the kernel dynamically allocates a nonreserved port number for each connection. The kernel selects the port number from a range of values between the value of the inet subsystem attribute ipport_userreserved_min and the value of the ipport_userreserved attribute. If you use the default attribute values, the number of simultaneous outgoing connections is limited to 3976.

Performance Benefit

Increasing the number of ports provides more ports for TCP and UDP applications.

You can modify the ipport_userreserved attribute without rebooting the system.

When to Tune

If your system requires many outgoing ports, you may want to increase the value of the ipport_userreserved attribute.

Recommended Values

The default value of the ipport_userreserved attribute is 5000, which means that the default number of ports is 3976 (5000 minus 1024).

If your system is a proxy server (for example, a Squid caching server or a firewall system) with a load of more than 4000 simultaneous connections, increase the value of the ipport_userreserved attribute to the maximum value of 65000.

We do not recommend that you reduce the value of the ipport_userreserved attribute to a value that is less than 5000 or increase it to a value that is higher than 65000.

You can also modify the range of outgoing connection ports. See Section 10.2.5 for information.

10.2.5 Modifying the Range of Outgoing Connection Ports

When a TCP or UDP application creates an outgoing connection, the kernel dynamically allocates a nonreserved port number for each connection. The kernel selects the port number from a range of values between the value of the inet subsystem attribute ipport_userreserved_min and the value of the ipport_userreserved attribute. Using the default values for these attributes, the range of outgoing ports starts at 1024 and stops at 5000.

Performance Benefit and Tradeoff

Modifying the range of outgoing connections provides TCP and UDP applications with a specific range of ports.

You can modify the `ipport_userreserved_min` and `ipport_userreserved` attributes without rebooting the system.

When to Tune

If your system requires outgoing ports from a particular range, you can modify the values of the `ipport_userreserved_min` and `ipport_userreserved` attributes.

Recommended Values

The default value of the `ipport_userreserved_min` attribute is 1024. The default value of the `ipport_userreserved` is 5000. The maximum value of both attributes is 65000.

Do not reduce the `ipport_userreserved` attribute to a value that is less than 5000, or reduce the `ipport_userreserved_min` attribute to a value that is less than 1024.

10.2.6 Disabling PMTU Discovery

Packets transmitted between servers are fragmented into units of a specific size to ease transmission of the data over routers and small-packet networks, such as Ethernet networks. When the `inet` subsystem attribute `pmtu_enabled` is enabled (set to 1, which is the default behavior), the system determines the largest common path maximum transmission unit (PMTU) value between servers and uses it as the unit size. The system also creates a routing table entry for each client network that attempts to connect to the server.

Performance Benefit and Tradeoff

If a server handles traffic among many remote clients, disabling the use of PMTU discovery can decrease the size of the kernel routing table, which improves server efficiency. However, on a server that handles local traffic and some remote traffic, disabling the use of PMTU discovery can degrade bandwidth.

You can modify the `pmtu_enabled` attribute without rebooting the system.

When to Tune

Under some circumstances, such as the presence on the path of a fragmenting FDDI-to-Ethernet bridge, the use of PMTU discovery can create a situation in which packets larger than a certain size disappear and makes

some data transfers impossible. In this case, disable the use of PMTU discovery. Also consider disabling the use of PMTU discovery if you have a server that handles traffic among many remote clients, or if you have an Internet server that has poor performance and the routing table increases to more than 1000 entries.

10.2.7 Increasing the Number of IP Input Queues

The `inet` subsystem attribute `ipqs` specifies the number of IP input queues.

Performance Benefit and Tradeoff

Increasing the number of IP input queues can reduce lock contention at the queue by increasing the number of queues and distributing the load.

You cannot modify the `ipqs` attribute without rebooting the system.

When to Tune

Increase the number of IP input queues if you have an SMP system that is an Internet server.

Recommended Values

For SMP systems that are Internet servers, increase the value of the `ipqs` attribute to 16. The maximum value is 64.

We recommend that you make the value of the `ipqs` attribute the same as the value of the `inet` subsystem attribute `tcbhashnum`. See Section 10.2.2 for more information.

10.2.8 Enabling mbuf Cluster Compression

The `socket` subsystem attribute `sbcompress_threshold` controls whether `mbuf` clusters are compressed at the socket layer. By default, `mbuf` clusters are not compressed (the `sbcompress_threshold` is set to 0).

Performance Benefit

Compressing `mbuf` clusters can prevent proxy servers from consuming all the available `mbuf` clusters.

You can modify the `sbcompress_threshold` attribute without rebooting the system.

When to Tune

You may want to enable `mbuf` cluster compression if you have a proxy server. These systems are more likely to consume all the available `mbuf` clusters if they are using FDDI instead of Ethernet.

To determine the memory that is being used for mbuf clusters, use the netstat -m command. The following example is from a firewall server with 128 MB of memory that does not have mbuf cluster compression enabled:

```
# netstat -m
  2521 Kbytes for small data mbufs (peak usage 9462 Kbytes)
 78262 Kbytes for mbuf clusters (peak usage 97924 Kbytes)
  8730 Kbytes for sockets (peak usage 14120 Kbytes)
  9202 Kbytes for protocol control blocks (peak usage 14551
     2 Kbytes for routing table (peak usage 2 Kbytes)
     2 Kbytes for socket names (peak usage 4 Kbytes)
     4 Kbytes for packet headers (peak usage 32 Kbytes)
 39773 requests for mbufs denied
     0 calls to protocol drain routines
 98727 Kbytes allocated to network
```

The previous example shows that 39773 requests for memory were denied. This indicates a problem because this value should be 0. The example also shows that 78 MB of memory has been assigned to mbuf clusters, and that 98 MB of memory is being consumed by the network subsystem.

Recommended Values

To enable mbuf cluster compression, modify the default value of the socket subsystem attribute sbcompress_threshold. Packets will be copied into the existing mbuf clusters if the packet size is less than this value. For proxy servers, specify a value of 600.

If you increase the value of the sbcompress_threshold attribute to 600, the memory allocated to the network subsystem immediately decreases to 18 MB, because compression at the kernel socket buffer interface results in a more efficient use of memory.

10.2.9 Enabling TCP Keepalive Functionality

Keepalive functionality enables the periodic transmission of messages on a connected socket to keep connections active. Sockets that do not exit cleanly are cleaned up when the keepalive interval expires. If keepalive is not enabled, those sockets will continue to exist until you reboot the system.

Applications enable keepalive for sockets by setting the setsockopt function's SO_KEEPALIVE option. To override programs that do not set keepalive on their own, or if you do not have access to the application sources, use the inet subsystem attribute tcp_keepalive_default to enable keepalive functionality.

Performance Benefit

Keepalive functionality cleans up sockets that do not exit cleanly when the keepalive interval expires.

You can modify the tcp_keepalive_default attribute without rebooting the system. However, sockets that already exist will continue to use old behavior, until the applications are restarted.

When to Tune

Enable keepalive if you require this functionality, and you do not have access to the source code.

Recommended Values

To override programs that do not set keepalive on their own, or if you do not have access to application source code, set the inet subsystem attribute tcp_keepalive_default to 1 to enable keepalive for all sockets.

If you enable keepalive, you can also configure the following inet subsystem attributes for sockets:

- The tcp_keepidle attribute specifies the amount of idle time before sending a keepalive probe (specified in 0.5-second units). The default interval is 2 hours.
- The tcp_keepintvl attribute specifies the amount of time (in 0.5-second units) between the retransmission of keepalive probes. The default interval is 75 seconds.
- The tcp_keepcnt attribute specifies the maximum number of keepalive probes that are sent before the connection is dropped. The default is 8 probes.
- The tcp_keepinit attribute specifies the maximum amount of time before an initial connection attempt times out in 0.5 second units. The default is 75 seconds.

Applications can modify the value of these attributes on a per-socket basis by using the setsockopt() call.

10.2.10 Improving the Lookup Rate for IP Addresses

The inet subsystem attribute inifaddr_hsize specifies the number of hash buckets in the kernel interface alias table (in_ifaddr).

If a system is used as a server for many different server domain names, each of which are bound to a unique IP address, the code that matches arriving packets to the right server address uses the hash table to speed lookup operations for the IP addresses.

Performance Benefit and Tradeoff

Increasing the number of hash buckets in the table can improve performance on systems that use large numbers of aliases.

You can modify the inifaddr_hsize attribute without rebooting the system.

When to Tune

Increase the number of hash buckets in the kernel interface alias table if your system uses large numbers of aliases.

Recommended Values

The default value of the inet subsystem attribute inifaddr_hsize is 32; the maximum value is 512.

For the best performance, the value of the inifaddr_hsize attribute is always rounded down to the nearest power of 2. If you are using more than 500 interface IP aliases, specify the maximum value of 512. If you are using less than 250 aliases, use the default value of 32.

10.2.11 Decreasing the TCP Partial-Connection Timeout Limit

The inet subsystem attribute tcp_keepinit specifies the amount of time that a partially established TCP connection remains on the socket listen queue before it times out. Partial connections consume listen queue slots and fill the queue with connections in the SYN_RCVD state.

Performance Benefit and Tradeoff

You can make partial connections time out sooner by decreasing the value of the tcp_keepinit attribute.

You can modify the tcp_keepinit attribute without rebooting the system.

When to Tune

You do not need to modify the TCP partial-connection timeout limit, unless the value of the somaxconn_drops attribute often increases. If this occurs, you may want to decrease the value of the tcp_keepinit attribute.

Recommended Values

The value of the tcp_keepinit attribute is in units of 0.5 seconds. The default value is 150 units (75 seconds). If the value of the sominconn attribute is 65535, use the default value of the tcp_keepinit attribute.

Do not set the value of the tcp_keepinit attribute too low, because you may prematurely break connections associated with clients on network paths that are slow or network paths that lose many packets. Do not set the value to less than 20 units (10 seconds).

10.2.12 Decreasing the TCP Connection Context Timeout Limit

The TCP protocol includes a concept known as the Maximum Segment Lifetime (MSL). When a TCP connection enters the TIME_WAIT state, it must remain in this state for twice the value of the MSL, or else undetected data errors on future connections can occur. The inet subsystem attribute tcp_msl determines the maximum lifetime of a TCP segment and the timeout value for the TIME_WAIT state.

Performance Benefit and Tradeoff

You can decrease the value of the tcp_msl attribute to make the TCP connection context time out more quickly at the end of a connection. However, this will increase the chance of data corruption.

You can modify the tcp_msl attribute without rebooting the system.

When to Tune

Usually, you do not have to modify the TCP connection context timeout limit.

Recommended Values

The value of the tcp_msl attribute is set in units of 0.5 seconds. The default value is 60 units (30 seconds), which means that the TCP connection remains in the TIME_WAIT state for 60 seconds (or twice the value of the MSL). In some situations, the default timeout value for the TIME_WAIT state (60 seconds) is too large, so reducing the value of the tcp_msl attribute frees connection resources sooner than the default behavior.

Do not reduce the value of the tcp_msl attribute unless you fully understand the design and behavior of your network and the TCP protocol. We recommend that you use the default value; otherwise, there is the potential for data corruption.

10.2.13 Decreasing the TCP Retransmission Rate

The inet subsystem attribute tcp_rexmit_interval_min specifies the minimum amount of time before the first TCP retransmission.

Performance Benefit and Tradeoff

You can increase the value of the tcp_rexmit_interval_min attribute to slow the rate of TCP retransmissions, which decreases congestion and improves performance.

You can modify the tcp_rexmit_interval_min attribute without rebooting the system.

When to Tune

Not every connection needs a long retransmission time. Usually, the default value is adequate. However, for some wide area networks (WANs), the default retransmission interval may be too small, causing premature retransmission timeouts. This may lead to duplicate transmission of packets and the erroneous invocation of the TCP congestion-control algorithms.

To check for retransmissions, use the `netstat -p tcp` command and examine the output for `data packets retransmitted`. See Section 2.4.5 for more information about the netstat command.

Recommended Values

The `tcp_rexmit_interval_min` attribute is specified in units of 0.5 seconds. The default value is 2 units (1 second).

Do not specify a value that is less than 1 unit. Do not change the attribute unless you fully understand TCP algorithms.

10.2.14 Disabling Delaying the Acknowledgment of TCP Data

By default, the system delays acknowledging TCP data. The `inet` subsystem attribute `tcpnodelack` determines whether the system delays acknowledging TCP data.

Performance Benefit and Tradeoff

Disabling delaying of TCP data may improve performance. However, this may adversely impact network bandwidth.

You can modify the `tcpnodelack` attribute without rebooting the system.

When to Tune

Usually, the default value of the `tcpnodelack` attribute is adequate. However, for some connections (for example, loopback), the delay can degrade performance. Use the `tcpdump` command to check for excessive delays.

Recommended Values

The default value of the `tcpnodelack` is 0. To disable the TCP acknowledgment delay, set the value of the `tcpnodelack` attribute to 1.

10.2.15 Increasing the Maximum TCP Segment Size

The `inet` subsystem attribute `tcp_mssdflt` specifies the TCP maximum segment size.

Performance Benefit and Tradeoff

Increasing the maximum TCP segment size allows sending more data per socket, but may cause fragmentation at the router boundary.

You can modify the tcp_mssdflt attribute without rebooting the system.

When to Tune

Usually you do not need to modify the maximum TCP segment size.

Recommended Values

The default value of the tcp_mssdflt attribute is 536. You can increase the value to 1460.

10.2.16 Increasing the Transmit and Receive Buffers for a UDP Socket

The inet subsystem attribute udp_sendspace specifies the default transmit buffer size for an Internet User Datagram Protocol (UDP) socket. The inet subsystem attribute udp_recvspace specifies the default receive buffer size for a UDP socket.

Performance Benefit and Tradeoff

Increasing the UDP transmit and receive socket buffers allows you to buffer more UDP packets per socket. However, increasing the values uses more memory when the buffers are being used by an application (sending or receiving data).

Note

UDP attributes do not affect network file system (NFS) performance.

You can modify the udp_sendspace and udp_recvspace attributes without rebooting the system. However, you must restart applications to use the new UDP socket buffer values.

When to Tune

Use the netstat -p udp command to check for full sockets. If the output shows many full sockets, increase the value of the udp_recvspace attribute.

Recommended Values

The default value of the udp_sendspace is 9 KB (9216 bytes). The default value of the udp_recvspace is 40 KB (42240 bytes). You can increase the values of these attributes to 64 KB.

10.2.17 Increasing the Maximum Size of a Socket Buffer

The socket subsystem attribute sb_max specifies the maximum size of a socket buffer.

Performance Benefit and Tradeoff

Increasing the maximum size of a socket buffer may improve performance if your applications can benefit from a large buffer size.

You can modify the sb_max attribute without rebooting the system.

When to Tune

If you require a large socket buffer, increase the maximum socket buffer size.

Recommended Values

The default value of the sb_max attribute is 128 KB. Increase this value before you increase the size of the transmit and receive socket buffers.

10.2.18 Preventing Dropped Input Packets

If the IP input queue overflows under a heavy network load, input packets may be dropped.

The inet subsystem attribute ipqmaxlen specifies the maximum length (in bytes) of the IP input queue (ipintrq) before input packets are dropped. The ifqmaxlen attribute specifies the number of output packets that can be queued to a network adapter before packets are dropped.

Performance Benefit and Tradeoff

Increasing the IP input queue can prevent packets from being dropped.

You can modify the ipqmaxlen and ifqmaxlen attributes without rebooting the system.

When to Tune

If your system drops packets, you may want to increase the values of the ipqmaxlen and ifqmaxlen attributes. To check for input dropped packets, examine the ipintrq kernel structure by using dbx. If the ifq_drops field is not 0, the system is dropping input packets. For example:

```
# dbx -k /vmunix
(dbx)print ipintrq
struct {
    ifq_head = (nil)
    ifq_tail = (nil)
    ifq_len = 0
    ifq_maxlen = 512
    ifq_drops = 128
 .
 .
```

.

Use the `netstat -id` command to monitor dropped output packets. Examine the output for a nonzero value in the `Drop` column for an interface. The following example shows 579 dropped output packets on the `tu1` network interface:

```
# netstat -id

Name  Mtu   Network  Address              Ipkts Ierrs    Opkts Oerrs  Coll Drop

fta0  4352  link     08:00:2b:b1:26:59    41586     0    39450     0     0   0
fta0  4352  DLI      none                 41586     0    39450     0     0   0
fta0  4352  10       fratbert             41586     0    39450     0     0   0
tu1   1500  link     00:00:f8:23:11:c8  2135983     0   163454    13  3376 579
tu1   1500  DLI      none               2135983     0   163454    13  3376 579
tu1   1500  red-net  ratbert            2135983     0   163454    13  3376 579
 .
 .
 .
```

In addition, you can use the `netstat -p ip`, and check for a nonzero number in the `lost packets due to resource problems` field or `no memory or interface queue was full` field. For example:

```
# netstat -p ip
ip:
        259201001 total packets received
        0 bad header checksums
        0 with size smaller than minimum
        0 with data size < data length
        0 with header length < data size
        0 with data length < header length
        25794050 fragments received
        0 fragments dropped (duplicate or out of space)
        802 fragments dropped after timeout
        0 packets forwarded
        67381376 packets not forwardable
                67381376 link-level broadcasts
        0 packets denied access
        0 redirects sent
        0 packets with unknown or unsupported protocol
        170988694 packets consumed here
        160039654 total packets generated here
        0 lost packets due to resource problems
        4964271 total packets reassembled ok
        2678389 output packets fragmented ok
        14229303 output fragments created
        0 packets with special flags set
```

Recommended Values

The default and minimum values for the `ipqmaxlen` and `ifqmaxlen` attributes are 1024; the maximum values are 65535. For most configurations, the default values are adequate. Only increase the values if you drop packets.

If your system drops packets, increase the values of the `ipqmaxlen` and `ifqmaxlen` attributes until you no longer drop packets. For example, you can increase the default values to 2000.

11

Managing File System Performance

To tune for better file-system performance, you must understand how your applications and users perform disk I/O, as described in Section 1.8, and how the file system you are using shares memory with processes, as described in Chapter 12. Using this information, you might improve file-system performance by changing the value of the kernel subsystem attributes described in this chapter.

This chapter describes how to tune:

- Caches used by file systems (Section 11.1)
- The Advanced File System (AdvFS) (Section 11.2)
- The UNIX file system (UFS) (Section 11.3)
- Network file system (NFS) (Section 11.4 and Chapter 5)

11.1 Tuning Caches

The kernel caches (temporarily stores) in memory recently accessed data. Caching data is effective because data is frequently reused and it is much faster to retrieve data from memory than from disk. When the kernel requires data, it checks if the data was cached. If the data was cached, it is returned immediately. If the data was not cached, it is retrieved from disk and cached. File-system performance is improved if data is cached and later reused.

Data found in a cache is called a **cache hit**, and the effectiveness of cached data is measured by a **cache hit rate**. Data that was not found in a cache is called a **cache miss**.

Cached data can be information about a file, user or application data, or metadata, which is data that describes an object (for example, a file). The following list identifies the types of data that are cached:

- A file name and its corresponding vnode is cached in the namei cache (Section 11.1.2).
- UFS user and application data and AdvFS user and application data and metadata are cached in the Unified Buffer Cache (UBC) (Section 11.1.3).
- UFS file metadata is cached in the metadata buffer cache (Section 11.1.4).

- AdvFS open file information is cached in access structures (Section 11.1.5).

11.1.1 Monitoring Cache Statistics

Table 11–1 describes the commands you can use to display and monitor cache information.

Table 11–1: Tools to Display Cache Information

Tools	Description	Reference
`(dbx) print` *`processor number`*	Displays namei cache statistics.	Section 11.1.2
`vmstat`	Displays virtual memory statistics.	Section 11.1.3 and Section 12.3.1
`(dbx) print bio_stats`	Displays metadata buffer cache statistics.	Section 11.3.2.3

11.1.2 Tuning the namei Cache

The virtual file system (VFS) presents to applications a uniform kernel interface that is abstracted from the subordinate file system layer. As a result, file access across different types of file systems is transparent to the user.

The VFS uses a structure called a **vnode** to store information about each open file in a mounted file system. If an application makes a read or write request on a file, VFS uses the vnode information to convert the request and direct it to the appropriate file system. For example, if an application makes a `read()` system call request on a file, VFS uses the vnode information to convert the system call to the appropriate type for the file system containing the file: `ufs_read()` for UFS, `advfs_read()` for AdvFS, or `nfs_read()` call if the file is in a file system mounted through NFS, then directs the request to the appropriate file system.

The VFS caches a recently accessed file name and its corresponding vnode in the **namei cache**. File-system performance is improved if a file is reused and its name and corresponding vnode are in the namei cache.

The following list describes the `vfs` subsystem attributes that relate to the namei cache:

Related Attributes

- `vnode_deallocation_enable` — Specifies whether or not to dynamically allocate vnode according to system demands.

  Value: 0 to 1

Default Value: 1 (enabled)

Disabling causes the operating system to use a static vnode pool. For the best performance, do not disable dynamic vnode allocation.

- `name_cache_hash_size` — Specifies the size, in slots, of the hash chain table for the namei cache.

  Default Value: 2 * (148 + 10 * maxusers) * 11 / 10 /15

- `vnode_age` — Specifies the amount of time, in seconds, before a free vnode can be recycled.

  Value: 0 to 2,147,483,647
  Default Value: 120 seconds

- `namei_cache_valid_time` — Specifies the amount of time, in seconds, that a namei cache entry can remain in the cache before it is discarded.

  Value: 0 to 2,147,483,647
  Default Value: 1200 (seconds) for 32-MB or larger systems; 30 (seconds) for 24-MB systems

Note

If you use increase the values of namei cache-related attributes, consider increasing file system attributes that cache file and directory information. If you use AdvFS, see Section 11.1.5 for more information. If you use UFS, see Section 11.1.4 for more information.

When to Tune

You can check namei cache statistics to see if you should change the values of namei cache related attributes. To check namei cache statistics, enter the `dbx print` command and specify a processor number to examine the `nchstats` data structure. For example:

```
# /usr/ucb/dbx –k /vmunix /dev/mem
(dbx) print processor_ptr[0].nchstats
```

Information similar to the following is displayed:

```
struct {
        ncs_goodhits = 18984
        ncs_neghits = 358
        ncs_badhits = 113
        ncs_falsehits = 23
        ncs_miss = 699
        ncs_long = 21
```

```
        ncs_badtimehits = 33
        ncs_collisions = 2
        ncs_unequaldups = 0
        ncs_newentry = 697
        ncs_newnegentry =  419
        ncs_gnn_hit = 1653
        ncs_gnn_miss = 12
        ncs_gnn_badhits = 12
        ncs_gnn_collision = 4
        ncs_pad = {
            [0] 0
        }
}
```

Table 11–2 describes when you might change the values of namei cache related attributes based on the `dbx print` output:

Table 11–2: When to Change the Values of the Namei Cache Related Attributes

If	Increase
The value of `ncs_goodhits` + `ncs_neghits` / `ncs_goodhits` + `ncs_neghits` + `ncs_miss` + `ncs_falsehits` is less than 80 percent	The value of either the `maxusers` attribute or the `name_cache_hash_size` attribute
The value of the `ncs_badtimehits` is more than 0.1 percent of the `ncs_goodhits`	The value of the `namei_cache_valid_time` attribute and the `vnode_age` attribute

You cannot modify the values of the `name_cache_hash_size` attribute, the `namei_cache_valid_time` attribute, or the `vnode_deallocation_enable` attribute without rebooting the system. You can modify the value of the `vnode_age` attribute without rebooting the system. See Chapter 3 for information about modifying subsystem attributes.

11.1.3 Tuning the UBC

The **Unified Buffer Cache (UBC)** shares with processes the memory that is not wired to cache UFS user and application data and AdvFS user and application data and metadata. File-system performance is improved if the data and metadata is reused and in the UBC.

Related Attributes

The following list describes the `vm` subsystem attributes that relate to the UBC:

- vm_ubcdirtypercent — Specifies the percentage of pages that must be dirty (modified) before the UBC starts writing them to disk.

  Value: 0 to 100
  Default Value: 10 percent

- ubc_maxdirtywrites — Specifies the number of I/O operations (per second) that the vm subsystem performs when the number of dirty (modified) pages in the UBC exceeds the value of the vm_ubcdirtypercent attribute.

  Value: 0 to 2,147,483,647
  Default Value: 5 (operations per second)

- ubc_maxpercent — Specifies the maximum percentage of physical memory that the UBC can use at one time.

  Value: 0 to 100
  Default Value: 100 percent

- ubc_borrowpercent — Specifies the percentage of memory above which the UBC is only borrowing memory from the vm subsystem. Paging does not occur until the UBC has returned all its borrowed pages.

  Value: 0 to 100
  Default Value: 20 percent
  Increasing this value might degrade system response time when a low-memory condition occurs (for example, a large process working set).

- ubc_minpercent — Specifies the minimum percentage of memory that the UBC can use. The remaining memory is shared with processes.

  Value: 0 to 100
  Default Value: 20 percent
  Increasing the value prevents large programs from completely consuming the memory that the UBC can use.
  For I/O servers, consider increasing the value to ensure that enough memory is available for the UBC.

- vm_ubcpagesteal — Specifies the minimum number of pages to be available for file expansion. When the number of available pages falls below this number, the UBC steals additional pages to anticipate the file's expansion demands.

  Value: 0 to 2,147,483,647
  Default Value: 24 (file pages)

- vm_ubcseqpercent — Specifies the maximum amount of memory allocated to the UBC that can be used to cache a single

  Value: 0 to 100
  Default Value: 10 percent of memory allocated to the UBC
  Consider increasing the value if applications write large files.

- vm_ubcseqstartpercent — Specifies a threshold value that determines when the UBC starts to recognize sequential file access and steal the UBC LRU pages for a file to satisfy its demand for pages. This value is the size of the UBC in terms of its percentage of physical memory.

  Value: 0 to 100
  Default Value: 50 percent
  Consider increasing the value if applications write large files.

Note

If the values of the ubc_maxpercent and ubc_minpercent attributes are close, you may degrade file system performance.

When to Tune

An insufficient amount of memory allocated to the UBC can impair file system performance. Because the UBC and processes share memory, changing the values of UBC-related attributes might cause the system to page. You can use the vmstat command to display virtual memory statistics that will help you to determine if you need to change values of UBC-related attributes. Table 11–3 describes when you might change the values UBC-related attributes based on the vmstat output:

Table 11–3: When to Change the Values of the UBC-Related Attributes

If vmstat Output Displays Excessive:	Action:
Paging but few or no page outs	Increase the value of the ubc_borrowpercent attribute.
Paging and swapping	Decrease the ubc_maxpercent attribute.

Table 11–3: When to Change the Values of the UBC-Related Attributes (cont.)

If vmstat Output Displays Excessive:	Action:
Paging	Force the system to reuse pages in the UBC instead of from the free list by making the value of the `ubc_maxpercent` attribute greater than the value of the `vm_ubseqstartpercent` attribute, which it is by default, and that the value of the `vm_ubcseqpercent` attribute is greater than a referenced file.
Page outs	Increase the value of the `ubc_min-percent` attribute.

See Section 12.3.1 for information on the `vmstat` command. See Section 12.1.2.2 for information about UBC memory allocation.

You can modify the value of any of the UBC parameters described in this section without rebooting the system. See Chapter 3 for information about modifying subsystem attributes.

Note

The performance of an application that generates a lot of random I/O is not improved by a large UBC, because the next access location for random I/O cannot be predetermined.

11.1.4 Tuning the Metadata Buffer Cache

At boot time, the kernel wires a percentage of memory for the metadata buffer cache. UFS file metadata, such as superblocks, inodes, indirect blocks, directory blocks, and cylinder group summaries are cached in the metadata buffer cache. File-system performance is improved if the metadata is reused and in the metadata buffer cache.

Related Attributes

The following list describes the `vfs` subsystem attributes that relate to the metadata buffer cache:

- `bufcache` — Specifies the size, as a percentage of memory, that the kernel wires for the metadata buffer cache.

  Value: 0 to 50
  Default value: 3 percent for 32-MB or larger systems and 2 percent for 24-MB systems

- `buffer_hash_size` — Specifies the size, in slots, of the hash chain table for the metadata buffer cache.

  Value: 0 to 524,287
  Default value: 2048 (slots)
  Increasing this value distributes the buffers to make the average chain lengths shorter, which improves UFS performance, but will reduce the amount of memory available to processes and the UBC.

  You cannot modify the values of the `buffer_hash_size` attribute or the `bufcache` attribute without rebooting the system. See Chapter 3 for information about modifying kernel subsystem attributes.

When to Tune

Consider increasing the size of the `bufcache` attribute if you have a high cache miss rate (low hit rate).

To determine if you have a high cache miss rate, use the `dbx print` command to display the `bio_stats` data structure. If the miss rate (block misses divided by the sum of the block misses and block hits) is more than 3 percent, consider increasing the value of the `bufcache` attribute. See Section 11.3.2.3 for more information on displaying the `bio_stats` data structure.

Note that increasing the value of the `bufcache` attribute will reduce the amount of memory available to processes and the UBC.

11.1.5 Tuning AdvFS Access Structures

At boot time, the system reserves a portion of the physical memory that is not wired by the kernel for AdvFS access structures. AdvFS caches information about open files and information about files that were opened but are now closed in AdvFS access structures. File-system performance is improved if the file information is reused and in an access structure.

AdvFS access structures are dynamically allocated and deallocated according to the kernel configuration and system demands.

Related Attribute

- `AdvfsAccessMaxPercent` — specifies, as a percentage, the maximum amount of pageable memory that can be allocated for AdvFS access structures.

  Value: 5 to 95
  Default value: 25 percent

You can modify the value of the AdvfsAccessMaxPercent attribute without rebooting the system. See Chapter 3 for information about modifying kernel subsystem attributes.

When to Tune

If users or applications reuse AdvFS files (for example, a proxy server), consider increasing the value of the AdvfsAccessMaxPercent attribute to allocate more memory for AdvFS access structures. Note that increasing the value of the AdvfsAccessMaxPercent attribute reduces the amount of memory available to processes and might cause excessive paging and swapping. You can use the vmstat command to display virtual memory statistics that will help you to determine excessive paging and swapping. See Section 12.3.1 for information on the vmstat command

Consider decreasing the amount of memory reserved for AdvFS access structures if:

- You do not use AdvFS.
- Your workload does not frequently open, close, and reopen the same files.
- You have a large-memory system (because the number of open files does not scale with the size of system memory as efficiently as UBC memory usage and process memory usage).

11.2 Tuning AdvFS

This section describes how to tune Advanced File System (AdvFS) queues, AdvFS configuration guidelines, and commands that you can use to display AdvFS information.

See the *AdvFS Administration* manual for information about AdvFS features and setting up and managing AdvFS.

11.2.1 AdvFS Configuration Guidelines

The amount of I/O contention on the volumes in a file domain is the most critical factor for fileset performance. This can occur on large, very busy file domains. To help you determine how to set up filesets, first identify:

- Frequently accessed data
- Infrequently accessed data
- Specific types of data (for example, temporary data or database data)
- Data with specific access patterns (for example, create, remove, read, or write)

Then, use the previous information and the following guidelines to configure filesets and file domains:

- Configure filesets that contain similar types of files in the same file domain to reduce disk fragmentation and improve performance. For example, do not place small temporary files, such as the output from `cron` and from news, mail, and Web cache servers, in the same file domain as a large database file.
- For applications that perform many file create or remove operations, configure multiple filesets and distribute files across the filesets. This reduces contention on individual directories, the root tag directory, quota files, and the frag file.
- Configure filesets used by applications with different I/O access patterns (for example, create, remove, read, or write patterns) in the same file domain. This might help to balance the I/O load.
- To reduce I/O contention in a multivolume file domain with more than one fileset, configure multiple domains and distribute the filesets across the domains. This enables each volume and domain transaction log to be used by fewer filesets.
- Filesets with a very large number of small files can affect `vdump` and `vrestore` commands at times. Using multiple filesets enables the `vdump` command to be run simultaneously on each fileset, and decreases the amount of time needed to recover filesets with the `vrestore` command.

Table 11–4 lists additional AdvFS configuration guidelines and performance benefits and tradeoffs. See the *AdvFS Administration* manual for more information about AdvFS.

Table 11–4: AdvFS Configuration Guidelines

Benefit	Guideline	Tradeoff
Data loss protection	Use LSM or RAID to store data using RAID1 (mirror data) or RAID5 (Section 11.2.1.1)	Requires LSM or RAID
Data loss protection	Force synchronous writes or enable atomic write data logging on a file (Section 11.2.1.2)	Might degrade file system performance
Improve performance for applications that read or write data only once	Enable direct I/O (Section 11.2.1.3)	Degrades performance of applications that repeatedly access the same data
Improve performance	Use AdvFS to distribute files in a file domain (Section 11.2.1.4)	None

Table 11–4: AdvFS Configuration Guidelines (cont.)

Benefit	Guideline	Tradeoff
Improve performance	Stripe data (Section 11.2.1.5)	None if using AdvFS or requires LSM or RAID
Improve performance	Defragment file domains (Section 11.2.1.6)	None
Improve performance	Decrease the I/O transfer size (Section 11.2.1.7)	None
Improves performance	Move the transaction log to a fast or uncongested disk (Section 11.2.1.8)	Might require an additional disk

The following sections describe these guidelines in more detail.

11.2.1.1 Storing Data Using RAID1 or RAID5

You can use LSM or hardware RAID to implement a RAID1 or RAID5 data storage configuration.

In a RAID1 configuration, LSM or hardware RAID stores and maintain mirrors (copies) of file domain or transaction log data on different disks. If a disk fails, LSM or hardware RAID uses a mirror to make the data available.

In a RAID5 configuration, LSM or hardware RAID stores parity information and data. If a disk fails, LSM or hardware RAID use the parity information and data on the remaining disks to reconstruct the missing data.

See the *Logical Storage Manager* manual for more information about LSM. See your storage hardware documentation for more information about hardware RAID.

11.2.1.2 Forcing a Synchronous Write Request or Enabling Persistent Atomic Write Data Logging

AdvFS writes data to disk in 8-KB units. By default, AdvFS asynchronous write requests are cached in the UBC, and the `write` system call returns a success value. The data is written to disk at a later time (asynchronously). AdvFS does not guarantee that all or part of the data will actually be written to disk if a crash occurs during or immediately after the write. For example, if the system crashes during a write that consists of two 8-KB units of data, only a portion (less than 16 KB) of the total write might have succeeded. This can result in partial data writes and inconsistent data.

You can configure AdvFS to force the write request for a specified file to be synchronous to ensure that data is successfully written to disk before the `write` system call returns a success value.

Enabling persistent atomic write data logging for a specified file writes the data to the transaction log file before it is written to disk. If a system crash occurs during or immediately after the `write` system call, the data in the log file is used to reconstruct the `write` system call upon recovery.

You cannot enable both forced synchronous writes and persistent atomic write data logging on a file. However, you can enable atomic write data logging on a file and also open the file with an `O_SYNC` option. This ensures that the write is synchronous, but also prevents partial writes if a crash occurs before the `write` system call returns.

To force synchronous write requests, enter:

```
# chfile -l on filename
```

A file that has persistent atomic write data logging enabled cannot be memory mapped by using the `mmap` system call, and it cannot have direct I/O enabled (see Section 11.2.1.3). To enable persistent atomic write data logging, enter:

```
# chfile -L on filename
```

A file that has persistent atomic write data logging will only be atomic if the writes are 8192 bytes or less. If the writes are greater than 8192 bytes, they are written in segments that are at most 8192 bytes in length with each segement an atomic-write.

To enable atomic-write data logging on AdvFS files that are NFS mounted, ensure that:

- The NFS property list daemon, `proplistd`, is running on the NFS server and that the fileset is mounted on the client by using the `mount` command and the `proplist` option.
- The offset into the file is on an 8-KB page boundary, because NFS performs I/O on 8-KB page boundaries. In this case, only 8192 byte segment that start on 8–KB page boundaries can be automatically written.

See `chfile`(8) and the *AdvFS Administration* manual for more information.

11.2.1.3 Enabling Direct I/O

You can enable direct I/O to significantly improve disk I/O throughput for applications that do not frequently reuse previously accessed data. The following lists considerations if you enable direct I/O:

- Data is not cached in the UBC and reads and writes are synchronous. You can use the asynchronous I/O (AIO) functions (`aio_read` and `aio_write`) to enable an application to achieve an asynchronous-like

behavior by issuing one or more synchronous direct I/O requests without waiting for their completion.

- Although direct I/O supports I/O requests of any byte size, the best performance occurs when the requested byte transfer is aligned on a disk sector boundary and is an even multiple of the underlying disk sector size.

You cannot enable direct I/O for a file if it is already opened for data logging or if it is memory mapped. Use the `fcntl` system call with the `F_GETCACHEPOLICY` argument to determine if an open file has direct I/O enabled.

To enable direct I/O for a specific file, use the `open` system call and set the `O_DIRECTIO` file access flag. A file remains opened for direct I/O until all users close the file.

See `fcntl`(2), `open`(2), the *AdvFS Administration* manual, and the *Programmer's Guide* for more information.

11.2.1.4 Using AdvFS to Distribute Files

If the files in a multivolume domain are not evenly distributed, performance might be degraded. You can distribute space evenly across volumes in a multivolume file domain to balance the percentage of used space among volumes in a domain. Files are moved from one volume to another until the percentage of used space on each volume in the domain is as equal as possible.

To determine if you need to balance files, enter:

```
# showfdmn file_domain_name
```

Information similar to the following is displayed:

```
               Id     Date Created       LogPgs Version   Domain Name
3437d34d.000ca710  Sun Oct 5 10:50:05 2001  512       3   usr_domain
 Vol  512-Blks   Free % Used  Cmode Rblks  Wblks  Vol Name
  1L   1488716 549232    63%     on   128    128  /dev/disk/dsk0g
  2     262144 262000     0%     on   128    128  /dev/disk/dsk4a
    --------- -------  ------
      1750860 811232    54%
```

The `% Used` field shows the percentage of volume space that is currently allocated to files or metadata (the fileset data structure). In the previous example, the `usr_domain` file domain is not balanced. Volume 1 has 63 percent used space while volume 2 has 0 percent used space (it was just added).

To distribute the percentage of used space evenly across volumes in a multivolume file domain, enter:

```
# balance file_domain_name
```

The `balance` command is transparent to users and applications, and does not affect data availability or split files. Therefore, file domains with very large files may not balance as evenly as file domains with smaller files and you might need to manually move large files into the same volume in a multivolume file domain.

To determine if you should move a file, enter:

```
# showfile -x file_name
```

Information similar to the following is displayed:

```
    Id Vol PgSz Pages XtntType  Segs  SegSz  I/O  Perf  File
8.8002   1   16    11   simple    **     ** async  18%  src

            extentMap: 1
        pageOff    pageCnt     vol    volBlock    blockCnt
              0          1       1      187296          16
              1          1       1      187328          16
              2          1       1      187264          16
              3          1       1      187184          16
              4          1       1      187216          16
              5          1       1      187312          16
              6          1       1      187280          16
              7          1       1      187248          16
              8          1       1      187344          16
              9          1       1      187200          16
             10          1       1      187232          16
        extentCnt: 11
```

The file in the previous example is a good candidate to move to another volume because it has 11 extents and an 18 percent performance efficiency as shown in the `Perf` field. A high percentage indicates optimal efficiency.

To move a file to a different volume in the file domain, enter:

```
# migrate [-p pageoffset] [-n pagecount] [-s volumeindex_from] \
[-d volumeindex_to] file_name
```

You can specify the volume from which a file is to be moved, or allow the system to pick the best space in the file domain. You can move either an entire file or specific pages to a different volume.

Note that using the `balance` utility after moving files might move files to a different volume.

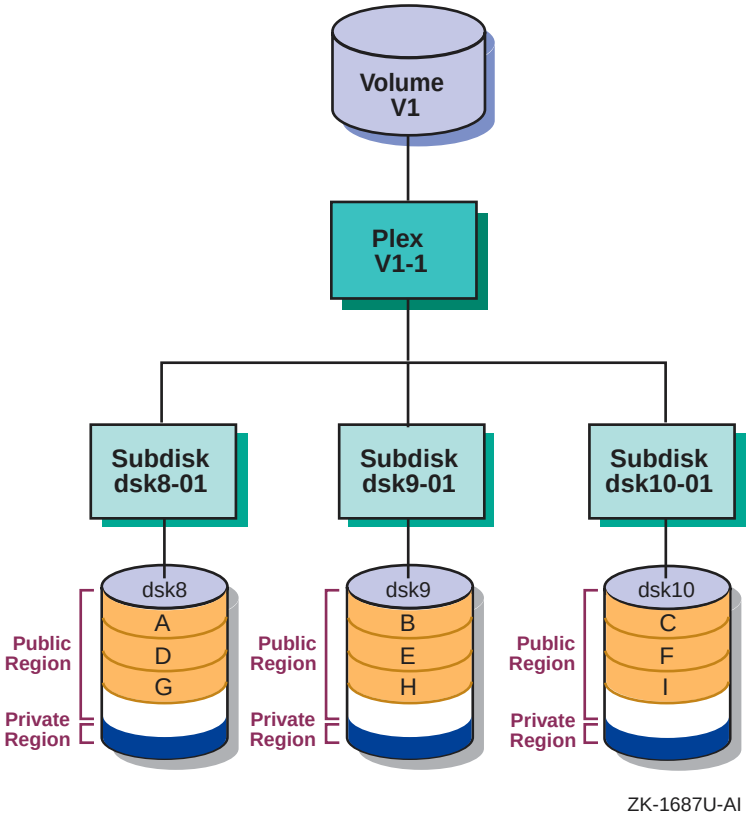
See `showfdmn`(8), `migrate`(8), and `balance`(8) for more information.

11.2.1.5 Striping Data

You can use AdvFS, LSM, or hardware RAID to **stripe** (distribute) data. Striped data is data that is separated into units of equal size, then written to two or more disks, creating a stripe of data. The data can be simultaneously written if there are two or more units and the disks are on different SCSI buses.

Figure 11–1 shows how a write request of 384 KB of data is separated into six 64-KB data units and written to three disks as two complete stripes.

Figure 11–1: Striping Data



Use only one method to stripe data. In some specific cases, using multiple striping methods can improve performance, but only if:

- Most of the I/O requests are large (greater than or equal to 1 MB)
- The data is striped over multiple RAID sets on different controllers
- The LSM or AdvFS stripe size is a multiple of the full hardware RAID stripe size

See `stripe`(8) for more information about using AdvFS to stripe data. See the *Logical Storage Manager* manual for more information about using LSM to stripe data. See your storage hardware documentation for more information about using hardware RAID to stripe data.

11.2.1.6 Defragmenting a File Domain

An **extent** is a contiguous area of disk space that AdvFS allocates to a file. Extents consist of one or more 8-KB pages. When storage is added to a file, it is grouped in extents. If all data in a file is stored in contiguous blocks, the file has one file extent. However, as files grow, contiguous blocks on the disk may not be available to accommodate the new data, so the file must be spread over discontiguous blocks and multiple file extents.

File I/O is most efficient when there are few extents. If a file consists of many small extents, AdvFS requires more I/O processing to read or write the file. Disk fragmentation can result in many extents and may degrade read and write performance because many disk addresses must be examined to access a file.

To display fragmentation information for a file domain, enter:

```
# defragment -vn file_domain_name
```

Information similar to the following is displayed:

```
defragment: Gathering data for 'staff_dmn'
Current domain data:
  Extents:                 263675
  Files w/ extents:        152693
  Avg exts per file w/exts:  1.73
  Aggregate I/O perf:          70%
  Free space fragments:     85574
              <100K   <1M   <10M   >10M
  Free space:   34%   45%    19%     2%
  Fragments:  76197  8930    440      7
```

Ideally, you want few extents for each file.

Although the `defragment` command does not affect data availability and is transparent to users and applications, it can be a time-consuming process and requires disk space. Run the `defragment` command during low file system activity as part of regular file system maintenance, or if you experience problems because of excessive fragmentation.

There is little performance benefit from defragmenting a file domain that contains files less than 8 KB, is used in a mail server, or is read-only.

You can also use the `showfile` command to check a file's fragmentation. See Section 11.2.2.4 and `defragment`(8) for more information.

11.2.1.7 Decreasing the I/O Transfer Size

AdvFS attempts to transfer data to and from the disk in sizes that are the most efficient for the device driver. This value is provided by the device driver and is called the **preferred transfer size**. AdvFS uses the preferred transfer size to:

- Consolidate contiguous, small I/O transfers into a larger, single I/O of the preferred transfer size. This results in a fewer number of I/O requests, which increases throughput.
- Prefetch, or read-ahead, as many subsequent pages for files being read sequentially up to the preferred transfer size in anticipation that those pages will eventually be read by the applicaton.

Generally, the I/O transfer size provided by the device driver is the most efficient. However, in some cases you may want to reduce the AdvFS I/O transfer size. For example, if your AdvFS fileset is using LSM volumes, the preferred transfer size might be very high. This could cause the cache to be unduly diluted by the buffers for the files being read. If this is suspected, reducing the read transfer size may alleviate the problem.

For systems with impaired `mmap` page faulting or with limited memory, limit the read transfer size to limit the amount of data that is prefetched; however, this will limit I/O consolidation for all reads from this disk.

To display the I/O transfer sizes for a disk, enter:

```
# chvol -l block_special_device_name domain
```

To modify the read I/O transfer size, enter:

```
# chvol -r blocks block_special_device_name domain
```

To modify the write I/O transfer size, enter:

```
# chvol -w blocks block_special_device_name domain
```

See `chvol`(8) for more information.

Each device driver has a minimum and maximum value for the I/O transfer size. If you use an unsupported value, the device driver automatically limits the value to either the largest or smallest I/O transfer size it supports. See your device driver documentation for more information on supported I/O transfer sizes.

11.2.1.8 Moving the Transaction Log

Place the AdvFS transaction log on a fast or uncongested disk and bus; otherwise, performance might be degraded.

To display volume information, enter:

```
# showfdmn file_domain_name
```

Information similar to the following is displayed:

```
               Id              Date Created  LogPgs  Domain Name
35ab99b6.000e65d2  Tue Jul 14 13:47:34 2002     512  staff_dmn

  Vol   512-Blks        Free  % Used  Cmode  Rblks  Wblks  Vol Name
   3L     262144      154512     41%     on    256    256  /dev/rz13a
```

```
  4      786432      452656     42%     on    256    256  /dev/rz13b
     ----------  ----------  ------
        1048576      607168     42%
```

In the `showfdmn` command display, the letter `L` displays next to the volume that contains the transaction log.

If the transaction log is located on a slow or busy disk, you can:

- Move the transaction log to a different disk.

  Use the `switchlog` command to move the transaction log.

- Divide a large multivolume file domain into several smaller file domains. This will distribute the transaction log I/O across multiple logs.

  To divide a multivolume domain into several smaller domains, create the smaller domains and then copy portions of the large domain into the smaller domains. You can use the AdvFS `vdump` and `vrestore` commands to allow the disks being used in the large domain to be used in the construction of the several smaller domains.

See `showfdmn`(8), `switchlog`(8), `vdump`(8), and `vrestore`(8) for more information.

11.2.2 Monitoring AdvFS Statistics

Table 11–5 describes the commands you can use to display AdvFS information.

Table 11–5: Tools to Display AdvFS Information

Tool	Description	Reference
`advfsstat`	Displays AdvFS performance statistics.	Section 11.2.2.1
`advscan`	Displays disks in a file domain.	Section 11.2.2.2
`showfdmn`	Displays information about AdvFS file domans and volumes.	Section 11.2.2.3
`showfsets`	Displays AdvFS fileset information for a file domain.	Section 11.2.2.5
`showfile`	Displays information about files in an AdvFS fileset.	Section 11.2.2.4

The following sections describe these commands in more detail.

11.2.2.1 Displaying AdvFS Performance Statistics

To display detailed information about a file domain, including use of the UBC and namei cache, fileset vnode operations, locks, bitfile metadata table (BMT) statistics, and volume I/O performance, use the `advfsstat` command.

The following example displays volume I/O queue statistics:

```
# advfsstat -v 3 [-i number_of_seconds] file_domain
```

Information, in units of one disk block (512 bytes), similar to the following is displayed:

```
  rd   wr   rg  arg   wg  awg  blk ubcr flsh  wlz  sms  rlz  con  dev
   0    0    0    0    0    0   1M    0  10K 303K  51K  33K  33K  44K
```

You can use the `-i` option to display information at specific time intervals, in seconds.

The previous example displays:

- `rd` (read) and `wr` (write) requests

  Compare the number of read requests to the number of write requests. Read requests are blocked until the read completes, but asynchronous write requests will not block the calling thread, which increases the throughput of multiple threads.

- `rg` and `arg` (consolidated reads) and `wg` and `awg` (consolidated writes)

  The consolidated read and write values indicate the number of disparate reads and writes that were consolidated into a single I/O to the device driver. If the number of consolidated reads and writes decreases compared to the number of reads and writes, AdvFS may not be consolidating I/O.

- `blk` (blocking queue), `ubcr` (ubc request queue), `flsh` (flush queue), `wlz` (wait queue), `sms` (smooth sync queue), `rlz` (ready queue), `con` (consol queue), and `dev` (device queue). See Section 11.2.3 for information on AdvFS I/O queues.

  If you are experiencing poor performance, and the number of I/O requests on the `flsh`, `blk`, or `ubcr` queues increases continually while the number on the `dev` queue remains fairly constant, the application may be I/O bound to this device. You might eliminate the problem by adding more disks to the domain or by striping with LSM or hardware RAID.

To display the number of file creates, reads, and writes and other operations for a specified domain or fileset, enter:

```
# advfsstat [-i number_of_seconds] -f 2 file_domain file_set
```

Information similar to the following is displayed:

```
  lkup  crt geta read writ fsnc dsnc   rm   mv rdir  mkd  rmd link
     0    0    0    0    0    0    0    0    0    0    0    0    0
     4    0   10    0    0    0    0    2    0    2    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0
    24    8   51    0    9    0    0    3    0    0    4    0    0
  1201  324 2985    0  601    0    0  300    0    0    0    0    0
  1275  296 3225    0  655    0    0  281    0    0    0    0    0
  1217  305 3014    0  596    0    0  317    0    0    0    0    0
  1249  304 3166    0  643    0    0  292    0    0    0    0    0
  1175  289 2985    0  601    0    0  299    0    0    0    0    0
   779  148 1743    0  260    0    0  182    0   47    0    4    0
     0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0
```

See advfsstat(8) for more information.

11.2.2.2 Displaying Disks in an AdvFS File Domain

Use the advscan command:

- To search all devices and LSM disk groups for AdvFS domains.
- To rebuild all or part of your /etc/fdmns directory if you deleted the /etc/fdmns directory, a directory domain under /etc/fdmns, or links from a domain directory under /etc/fdmns.
- If you moved devices in a way that has changed device numbers.

To display AdvFS volumes on devices or in an LSM disk group, enter:

```
# advscan device | LSM_disk_group
```

Information similar to the following is displayed:

```
Scanning disks  dsk0 dsk5
Found domains:
usr_domain
          Domain Id       2e09be37.0002eb40
          Created         Thu Jun 26 09:54:15 2002
          Domain volumes          2
          /etc/fdmns links        2
          Actual partitions found:
                                  dsk0c
                                  dsk5c
```

To re-create missing domains on a device, enter:

```
# advscan -r device
```

Information similar to the following is displayed:

```
Scanning disks  dsk6
Found domains: *unknown*
          Domain Id       2f2421ba.0008c1c0
          Created         Mon Jan 20 13:38:02 2002
          Domain volumes          1
          /etc/fdmns links        0
          Actual partitions found:
                                  dsk6a*
*unknown*
         Domain Id       2f535f8c.000b6860
```

```
        Created         Tue Feb 25 09:38:20 2002
        Domain volumes          1
        /etc/fdmns links        0
        Actual partitions found:
                                dsk6b*

Creating /etc/fdmns/domain_dsk6a/
       linking dsk6a
Creating /etc/fdmns/domain_dsk6b/
       linking dsk6b
```

See advscan(8) for more information.

11.2.2.3 Displaying AdvFS File Domains

To display information about a file domain, including the date created and the size and location of the transaction log, and information about each volume in the domain, including the size, the number of free blocks, the maximum number of blocks read and written at one time, and the device special file, enter:

```
# showfdmn file_domain
```

Information similar to the following is displayed:

```
             Id              Date Created  LogPgs  Version  Domain Name
34f0ce64.0004f2e0  Wed Mar 17 15:19:48 2002     512        4  root_domain

  Vol   512-Blks        Free  % Used  Cmode  Rblks  Wblks  Vol Name
   1L     262144       94896     64%     on    256    256  /dev/disk/dsk0a
```

For multivolume domains, the showfdmn command also displays the total volume size, the total number of free blocks, and the total percentage of volume space currently allocated.

See showfdmn(8) for more information about the output of the command.

11.2.2.4 Displaying AdvFS File Information

To display detailed information about files (and directories) in an AdvFS fileset, enter:

```
# showfile filename...
```

or

```
# showfile *
```

The * displays the AdvFS characteristics for all of the files in the current working directory.

Information similar to the following is displayed:

```
        Id  Vol  PgSz  Pages  XtntType  Segs  SegSz  I/O   Perf  File
  23c1.8001    1    16      1    simple    **     **  ftx   100%  OV
  58ba.8004    1    16      1    simple    **     **  ftx   100%  TT_DB
        **   **    **     **   symlink    **     **   **     **  adm
```

```
  239f.8001    1    16      1    simple    **     **  ftx   100%  advfs
         **   **    **     **   symlink    **     **   **     **  archive
     9.8001    1    16      2    simple    **     **  ftx   100%  bin (index)
         **   **    **     **   symlink    **     **   **     **  bsd
         **   **    **     **   symlink    **     **   **     **  dict
   288.8001    1    16      1    simple    **     **  ftx   100%  doc
   28a.8001    1    16      1    simple    **     **  ftx   100%  dt
         **   **    **     **   symlink    **     **   **     **  man
  5ad4.8001    1    16      1    simple    **     **  ftx   100%  net
         **   **    **     **   symlink    **     **   **     **  news
   3e1.8001    1    16      1    simple    **     **  ftx   100%  opt
         **   **    **     **   symlink    **     **   **     **  preserve
         **   **    **     **     advfs    **     **   **     **  quota.group
         **   **    **     **     advfs    **     **   **     **  quota.user
     b.8001    1    16      2    simple    **     **  ftx   100%  sbin (index)
         **   **    **     **   symlink    **     **   **     **  sde
   61d.8001    1    16      1    simple    **     **  ftx   100%  tcb
         **   **    **     **   symlink    **     **   **     **  tmp
         **   **    **     **   symlink    **     **   **     **  ucb
  6df8.8001    1    16      1    simple    **     **  ftx   100%  users
```

See `showfile`(8) for more information about the command output.

11.2.2.5 Displaying the AdvFS Filesets in a File Domain

To display information about the filesets in a file domain, including the fileset names, the total number of files, the number of used blocks, the quota status, and the clone status, enter:

showfsets *file_domain*

Information similar to the following is displayed:

```
usr
        Id           : 3d0f7cf8.000daec4.1.8001
        Files        :    30469,  SLim=        0,  HLim=        0
        Blocks (512) :  1586588,  SLim=        0,  HLim=        0
        Quota Status : user=off group=off
        Object Safety: off
        Fragging     : on
        DMAPI        : off
```

The previous example displays that a file domain called `dmn1` has one fileset and one clone fileset.

See `showfsets`(8) for more information.

11.2.3 Tuning AdvFS Queues

For each AdvFS volume, I/O requests are sent to one of the following queues:

- **Blocking, UBC request, and flush queue**

  The blocking, UBC request, and flush queues are queues in which reads and synchronous write requests are cached. A synchronous write request must be written to disk before it is considered complete and the application can continue.

The blocking queue is used primarily for reads and for kernel synchronous write requests. The UBC request queue is used for handling UBC requests to flush pages to disk. The flush queue is used primarily for buffer write requests, either through `fsync()`, `sync()`, or synchronous writes. Because the buffers on the blocking and UBC request queues are given slightly higher priority than those on the flush queue, kernel requests are handled more expeditiously and are not blocked if many buffers are waiting to be written to disk.

Processes that need to read or modify data in a buffer in the blocking, UBC request, or flush queue must wait for the data to be written to disk. This is in direct contrast with buffers on the lazy queues that can be modified at any time until they are finally moved down to the device queue.

- **Lazy queue**

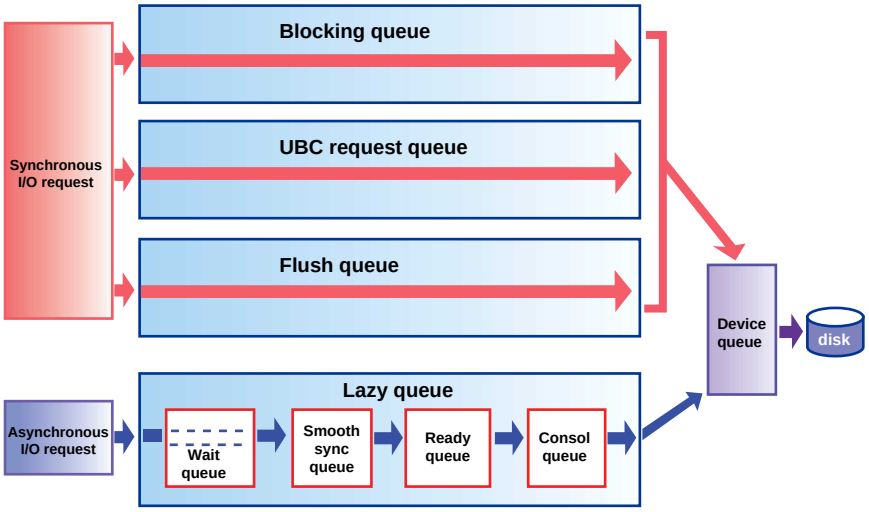
  The lazy queue is a logical series of queues in which asynchronous write requests are cached. When an asynchronous I/O request enters the lazy queue, it is assigned a timestamp. This timestamp is used to periodically flush the buffers down toward the disk in numbers large enough to allow them to be consolidated into larger I/Os. Processes can modify data in buffers at any time while they are on the lazy queue, potentially avoiding additonal I/Os. Descriptions of the queues in the lazy queue are provided after Figure 11–2.

All four queues (blocking, UBC request, flush, and lazy) move buffers to the **device queue**. As buffers are moved onto the device queue, logically contiguous I/Os are consolidated into larger I/O requests. This reduces the actual number of I/Os that must be completed. Buffers on the device queue cannot be modified until their I/O has completed.

The algorithms that move the buffers onto the device queue and favor taking buffers from the queues in the following order; blocking queue, UBC request queue, and then flush queue. All three are favored over the lazy queue. The size of the device queue is limited by device and driver resources. The algorithms that load the device queue use feedback from the drivers to know when the device queue is full. At that point the device is saturated and continued movement of buffers to the device queue would only degrade throughput to the device. The potential size of the device queue and how full it is, ultimately determines how long it may take to complete a synchronous I/O operation.

Figure 11–2 shows the movement of synchronous and asynchronous I/O requests through the AdvFS I/O queues.

Figure 11–2: AdvFS I/O Queues



Detailed descriptions of the AdvFS lazy queues are as follows:

- Wait queue — Asynchronous I/O requests that are waiting for an AdvFS transaction log write to complete first enter the wait queue. Each file domain has a transaction log that tracks fileset activity for all filesets in the file domain, and ensures AdvFS metadata consistency if a crash occurs.

  AdvFS uses write-ahead logging, which requires that when metadata is modified, the transaction log write must complete before the actual metadata is written. This ensures that AdvFS can always use the transaction log to create a consistent view of the file-system metadata. After the transaction log is written, I/O requests can move from the wait queue to the smooth sync queue.

- Smooth sync queue — Asynchronous I/O requests remain in the smooth sync queue for at least 30 seconds, by default. Allowing requests to remain in the smooth sync queue for a specified amount of time prevents I/O spikes, increases cache hit rates, and improves the consolidation of requests. After requests have aged in the smooth sync queue, they move to the ready queue.

- Ready queue — Asynchronous I/O requests are sorted in the ready queue. After the queue reaches a specified size, the requests are moved the consol queue.

- Consol queue — Asynchronous I/O requests are interleaved in the consol queue and moved to the device queue.

Related Attributes

The following list describes the vfs subsystem attributes that relate to AdvFS queues:

- smoothsync_age — Specifies the amount of time, in seconds, that a modified page ages before becoming eligible for the smoothsync mechanism to flush it to disk.

  Value: 0 to 60
  Default value: 30 seconds
  Setting the value to 0 sends data to the ready queue every 30 seconds, regardless of how long the data is cached.
  Increasing the value increases the chance of lost data if the system crashes, but can decrease net I/O load (improve performance) by allowing the dirty pages to remain cached longer.

  The smoothsync_age attribute is enabled when the system boots to multiuser mode and disabled when the system changes from multiuser mode to single-user mode. To permanently change the value of the smoothsync_age attribute, edit the following lines in the /etc/inittab file:

```
smsync:23:wait:/sbin/sysconfig -r vfs smoothsync_age=30 > /dev/null 2>&1
smsyncS:Ss:wait:/sbin/sysconfig -r vfs smoothsync_age=0 > /dev/null 2>&1
```

  You can use the smsync2 mount option to specify an alternate smoothsync policy that can further decrease the net I/O load. The default policy is to flush modified pages after they have been dirty for the smoothsync_age time period, regardless of continued modifications to the page. When you mount a filesystem using the smsync2 mount option, modified pages in nonmemory-mapped mode are not written to disk until they have been dirty and idle for the smoothsync_age time period.

  Note that AdvFS files in memory-mapped mode may not be flushed according to smoothsync_age.

- AdvfsSyncMmapPages — Specifies whether or not to disable smoothsync for applications that manage their own mmap page flushing.

  Value: 0 or 1
  Default value: 1 (enabled)

  See mmap(2) and msync(2) for more information.

- AdvfsReadyQLim — Specifies the size of the ready queue.

  Value: 0 to 32 K (blocks)
  Default value: 16 K (blocks)

You can modify the value of the AdvfsSyncMmapPages, smoothsync_age, and the AdvfsReadyQLim attributes without rebooting the system. See Chapter 3 for information about modifying kernel subsystem attributes.

When to Tune

If you reuse data, consider increasing:

- The amount of time I/O requests remain in the smoothsync queue to increase the possibility of a cache hit. However, doing so increases the chance that data might be lost if the system crashes.

  Use the advfsstat -S command to show cache statistics in the AdvFS smoothsync queue.

- The size of the ready queue to increase the possibility that I/O requests will be consolidated into a single, larger I/O and improve the possibility of a cache hit. However, doing so is not likely to have much influence if smoothsync is enabled and can increase the overhead in sorting the incoming requests onto the ready queue.

11.3 Tuning UFS

This section describes UFS configuration and tuning guidelines and commands that you can use to display UFS information.

11.3.1 UFS Configuration Guidelines

Table 11–6 lists UFS configuration guidelines and performance benefits and tradeoffs.

Table 11–6: UFS Configuration Guidelines

Benefit	Guideline	Tradeoff
Improve performance for small files	Make the file system fragment size equal to the block size (Section 11.3.1.1)	Wastes disk space for small files
Improve performance for large files	Use the default file system fragment size of 1 KB (Section 11.3.1.1)	Increases the overhead for large files
Free disk space and improve performance for large files	Reduce the density of inodes on a file system (Section 11.3.1.2)	Reduces the number of files that can be created
Improve performance for disks that do not have a read-ahead cache	Set rotational delay (Section 11.3.1.3)	None

Table 11–6: UFS Configuration Guidelines (cont.)

Benefit	Guideline	Tradeoff
Decrease the number of disk I/O operations	Increase the number of blocks combined for a cluster (Section 11.3.1.4)	None
Improve performance	Use a memory file system (MFS) (Section 11.3.1.5)	Does not ensure data integrity because of cache volatility
Control disk space usage	Use disk quotas (Section 11.3.1.6)	Might result in a slight increase in reboot time
Allow more mounted file systems	Increase the maximum number of UFS and MFS mounts (Section 11.3.1.7)	Requires additional memory resources

The following sections describe these guidelines in more detail.

11.3.1.1 Modifying the File System Fragment and Block Sizes

The UFS file system block size is 8 KB. The default fragment size is 1 KB. You can use the `newfs` command to modify the fragment size to 1024 KB, 2048 KB, 4096 KB, or 8192 KB when you create it.

Although the default fragment size uses disk space efficiently, it increases the overhead for files less than 96 KB. If the average file in a file system is less than 96 KB, you might improve disk access time and decrease system overhead by making the file-system fragment size equal to the default block size (8 KB).

See `newfs`(8) for more information.

11.3.1.2 Reducing the Density of inodes

An inode describes an individual file in the file system. The maximum number of files in a file system depends on the number of inodes and the size of the file system. The system creates an inode for each 4 KB (4096 bytes) of data space in a file system.

If a file system will contain many large files and you are sure that you will not create a file for each 4 KB of space, you can reduce the density of inodes on the file system. This will free disk space for file data, but also reduces the number of files that can be created.

To do this, use the `newfs -i` command to specify the amount of data space allocated for each inode when you create the file system. See `newfs`(8) for more information.

11.3.1.3 Set Rotational Delay

The UFS `rotdelay` parameter specifies the time, in milliseconds, to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive blocks in a file. By default, the `rotdelay` parameter is set to 0 to allocate blocks continuously. It is useful to set `rotdelay` on disks that do not have a read-ahead cache. For disks with cache, set the `rotdelay` to 0.

Use either the `tunefs` command or the `newfs` command to modify the `rotdelay` value.

See `newfs`(8) and `tunefs`(8) for more information.

11.3.1.4 Increasing the Number of Blocks Combined for a Cluster

The value of the UFS `maxcontig` parameter specifies the number of blocks that can be combined into a single cluster (or file-block group). The default value of `maxcontig` is 8. The file system attempts I/O operations in a size that is determined by the value of `maxcontig` multiplied by the block size (8 KB).

Device drivers that can chain several buffers together in a single transfer should use a `maxcontig` value that is equal to the maximum chain length. This may reduce the number of disk I/O operations.

Use the `tunefs` command or the `newfs` command to change the value of `maxcontig`.

See `newfs`(8) and `tunefs`(8) for more information.

11.3.1.5 Using MFS

The memory file system (MFS) is a UFS file system that resides only in memory. No permanent data or file structures are written to disk. An MFS can improve read/write performance, but it is a volatile cache. The contents of an MFS are lost after a reboot, unmount operation, or power failure.

Because no data is written to disk, an MFS is a very fast file system and can be used to store temporary files or read-only files that are loaded into the file system after it is created. For example, if you are performing a software build that would have to be restarted if it failed, use an MFS to cache the temporary files that are created during the build and reduce the build time.

See `mfs`(8) for more information.

11.3.1.6 Using UFS Disk Quotas

You can specify UFS file-system limits for user accounts and for groups by setting up UFS disk quotas, also known as UFS file system quotas. You can apply quotas to file-systems to establish a limit on the number of blocks and inodes (or files) that a user account or a group of users can allocate. You can set a separate quota for each user or group of users on each file system.

You may want to set quotas on file systems that contain home directories, because the sizes of these file systems can increase more significantly than other file systems. Do not set quotas on the `/tmp` file system.

Note that, unlike AdvFS quotas, UFS quotas may cause a slight increase in reboot time. See the *AdvFS Administration* manual for information about AdvFS quotas. See the *System Administration* manual for information about UFS quotas.

11.3.1.7 Increasing the Number of UFS and MFS Mounts

Mount structures are dynamically allocated when a mount request is made and subsequently deallocated when an unmount request is made.

Related Attributes

The `max_ufs_mounts` attribute specifies the maximum number of UFS and MFS mounts on the system.

Value: 0 to 2,147,483,647

Default value: 1000 (file system mounts)

You can modify the `max_ufs_mounts` attribute without rebooting the system. See Chapter 3 for information about modifying kernel subsystem attributes.

When to Tune

Increase the maximum number of UFS and MFS mounts if your system will have more than the default limit of 1000 mounts.

Increasing the maximum number of UFS and MFS mounts enables you to mount more file systems. However, increasing the maximum number mounts requires memory resources for the additional mounts.

11.3.2 Monitoring UFS Statistics

Table 11–7 describes the commands you can use to display UFS information.

Table 11–7: Tools to Display UFS Information

Tools	Decription	Reference
`dumpfs`	Displays UFS information.	Section 11.3.2.1
`(dbx) print ufs_clusterstats`	Displays UFS clustering statistics.	Section 11.3.2.2
`(dbx) print bio_stats`	Displays metadata buffer cache statistics.	Section 11.3.2.3

11.3.2.1 Displaying UFS Information

To display UFS information for a specified file system, including super block and cylinder group information, enter:

```
# dumpfs filesystem | /devices/disk/device_name
```

Information similar to the following is displayed:

```
 magic   11954   format  dynamic time   Tue Sep 14 15:46:52 2002
nbfree  21490   ndir    9       nifree  99541  nffree  60
ncg     65      ncyl    1027    size    409600  blocks  396062
bsize   8192    shift   13      mask    0xffffe000
fsize   1024    shift   10      mask    0xfffffc00
frag    8       shift   3       fsbtodb 1
cpg     16      bpg     798     fpg     6384    ipg     1536
minfree 10%     optim   time    maxcontig 8     maxbpg  2048
rotdelay 0ms    headswitch 0us  trackseek 0us   rps     60
```

The information contained in the first lines are relevant for tuning. Of specific interest are the following fields:

- `bsize` — The block size of the file system, in bytes (8 KB).
- `fsize` — The fragment size of the file system, in bytes. For the optimum I/O performance, you can modify the fragment size.
- `minfree` — The percentage of space that cannot be used by normal users (the minimum free space threshold).
- `maxcontig` — The maximum number of contiguous blocks that will be laid out before forcing a rotational delay; that is, the number of blocks that are combined into a single read request.
- `maxbpg` — The maximum number of blocks any single file can allocate out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. A large value for `maxbpg` can improve performance for large files.
- `rotdelay` — The expected time, in milliseconds, to service a transfer completion interrupt and initiate a new transfer on the same disk. It is

used to decide how much rotational spacing to place between successive blocks in a file. If `rotdelay` is 0, then blocks are allocated contiguously.

11.3.2.2 Monitoring UFS Clustering

To display how the system is performing cluster read and write transfers, use the `dbx print` command to examine the `ufs_clusterstats` data structure. For example:

```
# /usr/ucb/dbx –k /vmunix /dev/mem
(dbx) print ufs_clusterstats
```

Information similar to the following is displayed:

```
struct {
    full_cluster_transfers = 3130
    part_cluster_transfers = 9786
    non_cluster_transfers = 16833
    sum_cluster_transfers = {
        [0] 0
        [1] 24644
        [2] 1128
        [3] 463
        [4] 202
        [5] 55
        [6] 117
        [7] 36
        [8] 123
        [9] 0
         .
         .
         .
       [33]

    }
}
(dbx)
```

The previous example shows 24644 single-block transfers, 1128 double-block transfers, 463 triple-block transfers, and so on.

You can use the `dbx print` command to examine cluster reads and writes by specifying the `ufs_clusterstats_read` and `ufs_clusterstats_write` data structures respectively.

11.3.2.3 Displaying the Metadata Buffer Cache

To display statistics on the metadata buffer cache, including superblocks, inodes, indirect blocks, directory blocks, and cylinder group summaries, use the `dbx print` command to examine the `bio_stats` data structure. For example:

```
# /usr/ucb/dbx –k /vmunix /dev/mem
(dbx) print bio_stats
```

Information similar to the following is displayed:

```
struct {
    getblk_hits = 4590388
    getblk_misses = 17569
    getblk_research = 0
    getblk_dupbuf = 0
    getnewbuf_calls = 17590
    getnewbuf_buflocked = 0
    vflushbuf_lockskips = 0
    mntflushbuf_misses = 0
    mntinvalbuf_misses = 0
    vinvalbuf_misses = 0
    allocbuf_buflocked = 0
    ufssync_misses = 0
}
```

The number of block misses (`getblk_misses`) divided by the sum of block misses and block hits (`getblk_hits`) should not be more than 3 percent. If the number of block misses is high, you might want to increase the value of the `bufcache` attribute. See Section 11.1.4 for information on increasing the value of the `bufcache` attribute.

11.3.3 Tuning UFS for Performance

Table 11–8 lists UFS tuning guidelines and performance benefits and tradeoffs.

Table 11–8: UFS Tuning Guidelines

Benefit	Guideline	Tradeoff
Improve performance	Adjust UFS smoothsync and I/O throttling for asynchronous UFS I/O requests (Section 11.3.3.1)	None
Free CPU cycles and reduce the number of I/O operations	Delay UFS cluster writing (Section 11.3.3.2)	If I/O throttling is not used, might degrade real-time workload performance when buffers are flushed
Reduce the number of disk I/O operations	Increase the number of combined blocks for a cluster (Section 11.3.3.3)	Might require more memory to buffer data
Improve read and write performance	Defragment the file system (Section 11.3.3.4)	Requires down time

The following sections describe these guidelines in more detail.

11.3.3.1 Adjusting UFS Smooth Sync and I/O Throttling

UFS uses smoothsync and I/O throttling to improve UFS performance and to minimize system stalls resulting from a heavy system I/O load.

Smoothsync allows each dirty page to age for a specified time period before going to disk. This allows more opportunity for frequently modified pages to be found in the cache, which decreases the I/O load. Also, spikes in which large numbers of dirty pages are locked on the device queue are minimized because pages are enqueued to a device after having aged sufficiently, as opposed to getting flushed by the `update` daemon.

I/O throttling further addresses the concern of locking dirty pages on the device queue. It enforces a limit on the number of delayed I/O requests allowed to be on the device queue at any point in time. This allows the system to be more responsive to any synchronous requests added to the device queue, such as a read or the loading of a new program into memory. This can also decrease the amount and duration of process stalls for specific dirty buffers, as pages remain available until placed on the device queue.

Related Attributes

The `vfs` subsystem attributes that affect smoothsync and throttling are:

- `smoothsync_age` — Specifies the amount of time, in seconds, that a modified page ages before becoming eligible for the smoothsync mechanism to flush it to disk.

  Value: 0 to 60
  Default value: 30 seconds
  If set to 0, smoothsync is disabled and dirty page flushing is controlled by the `update` daemon at 30-second intervals.

  When to Tune

  Increasing the value increases the chance of lost data if the system crashes, but can decrease net I/O load (improve performance) by allowing the dirty pages to remain cached longer.

  The `smoothsync_age` attribute is enabled when the system boots to multiuser mode and disabled when the system changes from multiuser mode to single-user mode. To change the value of the `smoothsync_age` attribute, edit the following lines in the `/etc/inittab` file:

```
smsync:23:wait:/sbin/sysconfig -r vfs smoothsync_age=30 > /dev/null 2>&1
smsyncS:Ss:wait:/sbin/sysconfig -r vfs smoothsync_age=0 > /dev/null 2>&1
```

  You can use the `smsync2` mount option to specify an alternate smoothsync policy that can further decrease the net I/O load. The default policy is to flush modified pages after they have been dirty for the `smoothsync_age` time period, regardless of continued modifications to the page. When you mount a UFS using the `smsync2` mount option, modified pages are not written to disk until they have been dirty and idle for the `smoothsync_age` time period. Note that memory-mapped pages always use this default policy, regardless of the `smsync2` setting.

- io_throttle_shift — Specifies a value that limits the maximum number of concurrent delayed UFS I/O requests on an I/O device queue.

  Default value: 1 (2 seconds). However, the io_throttle_shift attribute only applies to file systems that you mount using the throttle mount option.

  The greater the number of requests on an I/O device queue, the longer it takes to process those requests and to make those pages and device available. The number of concurrent delayed I/O requests on an I/O device queue can be throttled (controlled) by setting the io_throttle_shift attribute. The calculated throttle value is based on the value of the io_throttle_shift attribute and the device's calculated I/O completion rate. The time required to process the I/O device queue is proportional to the throttle value. The correspondences between the value of the io_throttle_shift attribute and the time to process the device queue are:

Value of the io_throttle_shift Attribute	Time (in seconds) to Process Device Queue
-4	0.0625
-3	0.125
-2	0.25
-1	0.5
0	1
1	2
2	4
3	8
4	16

  Consider reducing the value of the io_throttle_shift attribute if your environment is particularly sensitive to delays in accessing the I/O device.

- io_maxmzthruput — Specifies whether or not to maximize I/O throughput or to maximize the availability of dirty pages. Maximizing I/O throughput works more aggressively to keep the device busy, but within the constraints of the io_throttle_shift attribute. Maximizing the availability of dirty pages favors decreasing the stall time experienced when waiting for dirty pages.

  Value: 0 (disabled) or 1 (enabled)

Default value: 1 (enabled). However, the io_throttle_maxmzthruput attribute only applies to file system that you mount using the throttle mount option.

When to Tune

Consider disabling the io_maxmzthruput attribute if your environment is particularly sensitive to delays in accessing sets of frequently used dirty pages or an environment in which I/O is confined to a small number of I/O-intensive applications, such that access to a specific set of pages becomes more important for overall performance than does keeping the I/O device busy.

You can modify the smoothsync_age, io_throttle_static, and io_throttle_maxmzthruput attributes without rebooting the system.

11.3.3.2 Delaying UFS Cluster Writing

By default, clusters of UFS pages are written asynchronously. You can configure clusters of UFS pages to be written delayed as other modified data and metadata pages are written.

Related Attribute

delay_wbuffers — Specifies whether or not clusters of UFS pages are written asynchronously or delayed.

Value: 0 or 1
Default value: 0 (asynchronously)
If the percentage of UBC dirty pages reaches the value of the delay_wbuffers_percent attribute, the clusters will be written asynchronously, regardless of the value of the delay_wbuffers attribute.

Delay writing clusters of UFS pages if your applications frequently write to previously written pages. This can result in a decrease in the total number of I/O requests. However, if you are not using I/O throttling, it might adversely affect real-time workload performance because the system will experience a heavy I/O load at sync time.

To delay writing clusters of UFS pages, use the dbx patch command to set the value of the delay_wbuffers kernel variable to 1 (enabled).

See Section 3.2 for information about using dbx.

11.3.3.3 Increasing the Number of Blocks in a Cluster

UFS combines contiguous blocks into clusters to decrease I/O operations. You can specify the number of blocks in a cluster.

Related Attribute

`cluster_maxcontig` — Specifies the number of blocks that are combined into a single I/O operation.

Default value: 32 blocks

If the specific file-system's rotational delay value is 0 (default), then UFS attempts to create clusters with up to *n* blocks, where *n* is either the value of the `cluster_maxcontig` attribute or the value from device geometry, whichever is smaller.

If the specific file-system's rotational delay value is nonzero, then *n* is the value of the `cluster_maxcontig` attribute, the value from device geometry, or the value of the `maxcontig` file-system attribute, whichever is smaller.

When to Tune

Increase the number of blocks combined for a cluster if your applications can use a large cluster size.

Use the `newfs` command to set the file-system rotational delay value and the value of the `maxcontig` attribute. Use the `dbx` command to set the value of the `cluster_maxcontig` attribute.

11.3.3.4 Defragmenting a File System

When a file consists of noncontiguous file extents, the file is considered fragmented. A very fragmented file decreases UFS read and write performance, because it requires more I/O operations to access the file.

When to Perform

Defragmenting a UFS file system improves file-system performance. However, it is a time-consuming process.

You can determine whether the files in a file system are fragmented by determining how effectively the system is clustering. You can do this by using the `dbx print` command to examine the `ufs_clusterstats` data structure. See Section 11.3.2.2 for information.

UFS block clustering is usually efficient. If the numbers from the UFS clustering kernel structures show that clustering is not effective, the files in the file system may be very fragmented.

Recommended Procedure

To defragment a UFS file system, follow these steps:

1. Back up the file system onto tape or another partition.

2. Create a new file system either on the same partition or a different partition.
3. Restore the file system.

See the *System Administration* manual for information about backing up and restoring data and creating UFS file systems.

11.4 Tuning NFS

The network file system (NFS) shares the Unified Buffer Cache (UBC) with the virtual memory subsystem and local file systems. NFS can put an extreme load on the network. Poor NFS performance is almost always a problem with the network infrastructure. Look for high counts of retransmitted messages on the NFS clients, network I/O errors, and routers that cannot maintain the load.

Lost packets on the network can severely degrade NFS performance. Lost packets can be caused by a congested server, the corruption of packets during transmission (which can be caused by bad electrical connections, noisy environments, or noisy Ethernet interfaces), and routers that abandon forwarding attempts too quickly.

For information about how to tune network file systems (NFS), see Chapter 5.

12

Managing Memory Performance

You may be able to improve Tru64 UNIX performance by optimizing your memory resources. Usually, the best way to improve performance is to eliminate or reduce paging and swapping. To do this, increase memory resources.

This chapter describes:

- How the operating system allocates virtual memory to processes and to file-system caches, and how memory is reclaimed (Section 12.1)
- How to configure swap space for high performance (Section 12.2)
- How to display information about memory usage (Section 12.3)
- The kernel subsystem attributes that you can modify to provide more memory resources to processes (Section 12.4)
- How to modify the paging and swapping operations (Section 12.5)
- How to reserve physical memory for shared memory (Section 12.6)
- How to use big pages to improve the performance of memory-intensize applications (Section 12.7)

12.1 Virtual Memory Operation

The operating system allocates physical memory in 8-KB units called pages. The virtual memory subsystem tracks and manages all the physical pages in the system and efficiently distributes the pages among three areas:

- **Static wired memory**

  Allocated at boot time and used for operating system data and text and for system tables, static wired memory is also used by the metadata buffer cache, which holds recently accessed UNIX file system (UFS) and CD-ROM file system (CDFS) metadata.

  You can reduce the amount of static wired memory only by removing subsystems or by decreasing the size of the metadata buffer cache (see Section 12.1.2.1).

- **Dynamically wired memory**

  Dynamically wired memory is allocated at boot time and used for dynamically allocated data structures, such as system hash tables.

User processes also allocate dynamically wired memory for address space by using virtual memory locking interfaces, including the `mlock` function. The amount of dynamically wired memory varies according to the demand. The `vm` subsystem attribute `vm_syswiredpercent` specifies the maximum amount of memory that a user process can wire (80 percent of physical memory, by default).

- **Physical memory for processes and data caching**

  Physical memory that is not wired is referred to as **pageable memory**. It is used for processes' most-recently accessed anonymous memory (modifiable virtual address space) and file-backed memory (memory that is used for program text or shared libraries). Pageable memory is also used to cache the most-recently accessed UFS file system data for reads and writes and for page faults from mapped file regions, in addition to AdvFS metadata and file data.

  The virtual memory subsystem allocates physical pages according to the process and file-system demand. Because processes and file systems compete for a limited amount of physical memory, the virtual memory subsystem periodically reclaims the oldest pages by writing their contents to swap space or disk (paging). Under heavy loads, entire processes may be suspended to free large amounts of memory (swapping). You can control virtual memory operation by tuning various `vm` subsystem attributes, as described in this chapter.

You must understand memory operation to determine which tuning guidelines will improve performance for your workload. The following sections describe how the virtual memory subsystem:

- Tracks physical pages (Section 12.1.1)
- Allocates memory to file system buffer caches (Section 12.1.2)
- Allocates memory to processes (Section 12.1.3)
- Reclaims pages (Section 12.1.4)

12.1.1 Physical Page Tracking

The virtual memory subsystem tracks all the physical memory pages in the system. Page lists are used to identify the location and age of each page. The oldest pages are the first to be reclaimed. At any one time, each physical page can be found on one of the following lists:

- **Wired list** — Pages that are wired and cannot be reclaimed
- **Free list** — Pages that are clean and are not being used

  Page reclamation begins when the size of the free list decreases to a tunable limit.

- **Active list** — Pages that are currently being used by processes or the Unified Buffer Cache (UBC)

  To determine which active pages to reclaim first, the page-stealer daemon identifies the oldest pages on the active list. The oldest pages that are being used by processes are designated as **inactive pages**. The oldest pages that are being used by the UBC are designated **UBC LRU** (Unified Buffer Cache least-recently used) pages.

Use the `vmstat` command to determine the number of pages that are on the page lists. Remember that pages on the active list (the `act` field in the `vmstat` output) include both inactive and UBC LRU pages.

12.1.2 File-System Buffer Cache Memory Allocation

The operating system uses caches to store file system user data and metadata. If the cached data is later reused, a disk I/O operation is avoided, which improves performance. This is because data can be retrieved from memory faster than a disk I/O operation.

The following sections describe these file-system caches:

- Metadata buffer cache (Section 12.1.2.1)
- Unified Buffer Cache (Section 12.1.2.2)

12.1.2.1 Metadata Buffer Cache Memory Allocation

At boot time, the kernel allocates wired memory for the metadata buffer cache. The cache acts as a layer between the operating system and disk by storing recently accessed UFS and CDFS metadata, which includes file header information, superblocks, inodes, indirect blocks, directory blocks, and cylinder group summaries. Performance is improved if the data is later reused and a disk operation is avoided.

The metadata buffer cache uses `bcopy` routines to move data in and out of memory. Memory in the metadata buffer cache is not subject to page reclamation.

The size of the metadata buffer cache is specified by the value of the `vfs` subsystem `bufcache` attribute. See Section 11.1.4 for information on tuning the `bufcache` attribute.

12.1.2.2 Unified Buffer Cache Memory Allocation

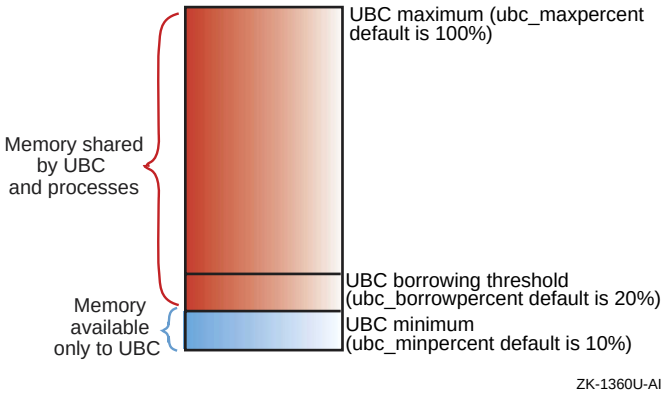
The physical memory that is not wired is available to processes and to the Unified Buffer Cache (UBC), which compete for this memory.

The UBC functions as a layer between the operating system and disk by storing recently accessed file-system data for reads and writes from

conventional file activity and holding page faults from mapped file sections. UFS caches user and application data in the UBC. AdvFS caches user and application data and metadata in the UBC. File-system performance is improved if the data and metadata is reused and in the UBC.

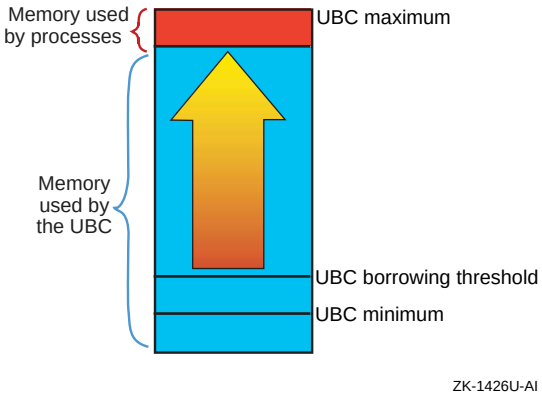
Figure 12–1 shows how the memory subsystem allocates physical memory to the UBC and for processes.

Figure 12–1: UBC Memory Allocation



At any one time, the amount of memory allocated to the UBC and to processes depends on file-system and process demands. For example, if file system activity is heavy and process demand is low, most of the pages will be allocated to the UBC, as shown in Figure 12–2.

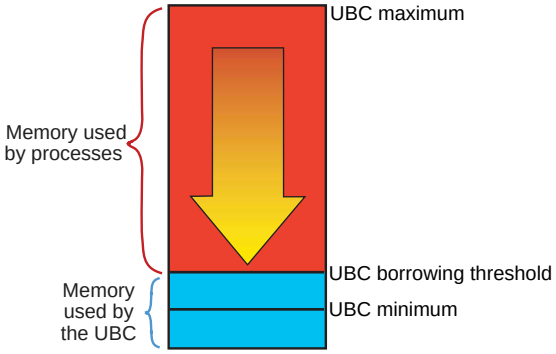
Figure 12–2: Memory Allocation During High File-System Activity and No Paging Activity



In contrast, heavy process activity, such as large increases in the working sets for large executables, will cause the memory subsystem to reclaim UBC

borrowed pages, down to the value of the `ubc_borrowpercent` attribute, as shown in Figure 12–3.

Figure 12–3: Memory Allocation During Low File-System Activity and High Paging Activity



The size of the UBC is specified by the value of the `vfs` subsystem UBC-related attributes. See Section 11.1.3 for information on tuning the UBC-related attribute.

12.1.3 Process Memory Allocation

Physical memory that is not wired is available to processes and the UBC, which compete for this memory. The virtual memory subsystem allocates memory resources to processes and to the UBC according to the demand, and reclaims the oldest pages if the demand depletes the number of available free pages.

The following sections describe how the virtual memory subsystem allocates memory to processes.

12.1.3.1 Process Virtual Address Space Allocation

The `fork` system call creates new processes. When you invoke a process, the `fork` system call:

1. Creates a UNIX process body, which includes a set of data structures that the kernel uses to track the process and a set of resource limitations. See `fork`(2) for more information.
2. Establishes a contiguous block of **virtual address space** for the process. Virtual address space is the array of virtual pages that the process can use to map into actual physical memory. Virtual address space is used for anonymous memory (memory that holds data elements

and structures that are modified during process execution) and for file-backed memory (memory used for program text or shared libraries).

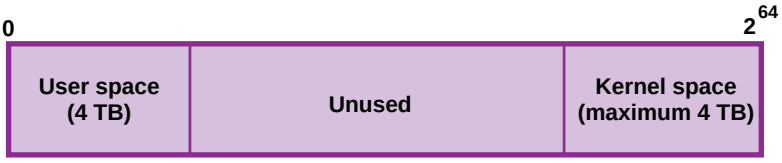
Because physical memory is limited, a process' entire virtual address space cannot be in physical memory at one time. However, a process can execute when only a portion of its virtual address space (its working set) is mapped to physical memory. Pages of anonymous memory and file-backed memory are paged in only when needed. If the memory demand increases and pages must be reclaimed, the pages of anonymous memory are paged out and their contents moved to swap space, while the pages of file-backed memory are simply released.

3. Creates one or more threads of execution. The default is one thread for each process. Multiprocessing systems support multiple process threads.

Although the virtual memory subsystem allocates a large amount of virtual address space for each process, it uses only part of this space. Only 4 TB is allocated for user space. User space is generally private and maps to a nonshared physical page. An additional 4 TB of virtual address space is used for kernel space. Kernel space usually maps to shared physical pages. The remaining space is not used for any purpose.

Figure 12–4 shows the use of process virtual address space.

Figure 12–4: Process Virtual Address Space Usage

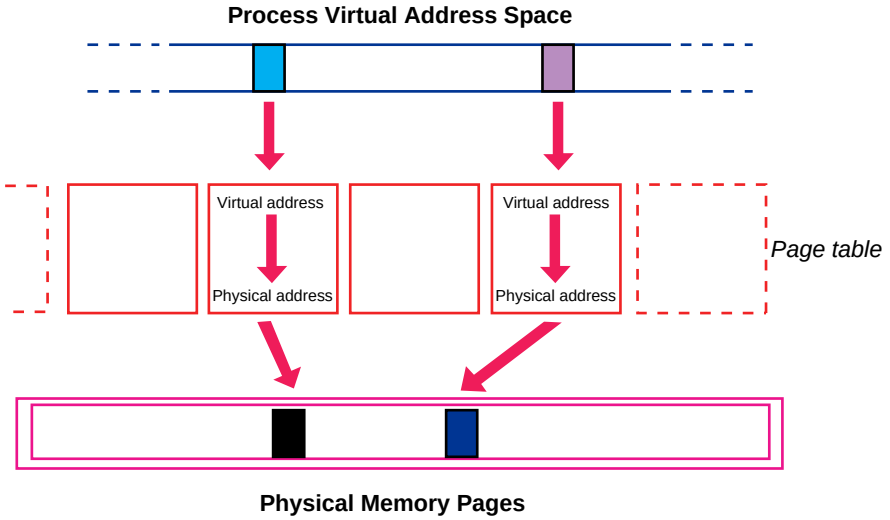


12.1.3.2 Virtual Address to Physical Address Translation

When a virtual page is touched (accessed), the virtual memory subsystem must locate the physical page and then translate the virtual address into a physical address. Each process has a **page table**, which is an array containing an entry for each current virtual-to-physical address translation. Page table entries have a direct relation to virtual pages (that is, virtual address 1 corresponds to page table entry 1) and contain a pointer to the physical page and protection information.

Figure 12–5 shows the translation of a virtual address into a physical address.

Figure 12–5: Virtual-to-Physical Address Translation



A process **resident set** is the complete set of all the virtual addresses that have been mapped to physical addresses (that is, all the pages that have been accessed during process execution). Resident set pages may be shared among multiple processes.

A process **working set** is the set of virtual addresses that are currently mapped to physical addresses. The working set is a subset of the resident set, and it represents a snapshot of the process resident set at one point in time.

12.1.3.3 Page Faults

When an anonymous (nonfile-backed) virtual address is requested, the virtual memory subsystem must locate the physical page and make it available to the process. This occurs at different speeds, depending on whether the page is in memory or on disk (see Figure 1–10).

If a requested address is currently being used (that is, the address is in the active page list), it will have an entry in the page table. In this case, the PAL code loads the physical address into the translation lookaside buffer, which then passes the address to the CPU. Because this is a memory operation, it occurs quickly.

If a requested address is not active in the page table, the PAL lookup code issues a **page fault**, which instructs the virtual memory subsystem to locate the page and make the virtual-to-physical address translation in the page table.

There are four different types of page faults:

1. If a requested virtual address is being accessed for the first time, a **zero-filled-on-demand page fault** occurs. The virtual memory subsystem performs the following tasks:

   a. Allocates an available page of physical memory.

   b. Fills the page with zeros.

   c. Enters the virtual-to-physical address translation in the page table.

2. If a requested virtual address has already been accessed and is located in the memory subsystem's internal data structures, a **short page fault** occurs. For example, if the physical address is located in the hash queue list or the page queue list, the virtual memory subsystem passes the address to the CPU and enters the virtual-to-physical address translation in the page table. This occurs quickly because it is a memory operation.

3. If a requested virtual address has already been accessed, but the physical page has been reclaimed, the page contents will be found either on the free page list or in swap space. If a page is located on the free page list, it is removed from the hash queue and the free list and then reclaimed. This operation occurs quickly and does not require disk I/O.

   If a page is found in swap space, a **page-in page fault** occurs. The virtual memory subsystem copies the contents of the page from swap space into the physical address and enters the virtual-to-physical address translation in the page table. Because this requires a disk I/O operation, it requires more time than a memory operation.

4. If a process needs to modify a read-only virtual page, a **copy-on-write page fault** occurs. The virtual memory subsystem allocates an available page of physical memory, copies the read-only page into the new page, and enters the translation in the page table.

The virtual memory subsystem uses the following techniques to improve process execution time and decrease the number of page faults:

- **Mapping additional pages**

  The virtual memory subsystem attempts to anticipate which pages the task will need next. Using an algorithm that checks which pages were most recently used, the number of available pages, and other factors, the subsystem maps additional pages along with the page that contains the requested address.

- **Page coloring**

  The virtual memory subsystem attempts to map a process' entire resident set into the secondary cache and executes the entire task, text, and data within the cache.

  The vm subsystem attribute private_cache_percent specifies the percentage of the secondary cache that is reserved for anonymous memory. This attribute is used only for benchmarking. The default is to reserve 50 percent of the cache for anonymous memory and 50 percent for file-backed memory (shared). To cache more anonymous memory, increase the value of the private_cache_percent attribute.

12.1.4 Page Reclamation

Because memory resources are limited, the virtual memory subsystem must periodically reclaim pages. The free page list contains clean pages that are available to processes and the UBC. As the demand for memory increases, the list may become depleted. If the number of pages falls below a tunable limit, the virtual memory subsystem will replenish the free list by reclaiming the least-recently used pages from processes and the UBC.

To reclaim pages, the virtual memory subsystem:

1. Prewrites modified pages to swap space in an attempt to forestall a memory shortage. See Section 12.1.4.1 for more information.
2. Begins paging if the demand for memory is not satisfied, as follows:

   a. Reclaims pages that the UBC has borrowed and puts them on the free list.

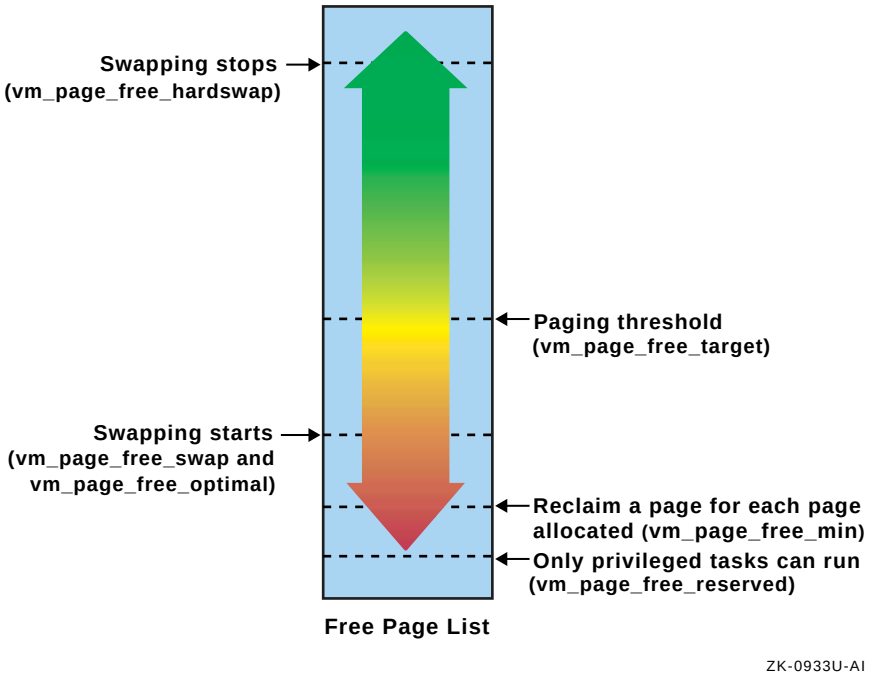
   b. Reclaims the oldest inactive and UBC LRU pages from the active page list, moves the contents of the modified pages to swap space or disk, and puts the clean pages on the free list.

   c. If needed, more aggressively reclaims pages from the active list.

   See Section 12.1.4.2 for more information about reclaiming memory by paging.
3. Begins swapping if the demand for memory is not met. The virtual memory subsystem temporarily suspends processes and moves entire resident sets to swap space, which frees large numbers of pages. See Section 12.1.4.3 for information about swapping.

The point at which paging and swapping start and stop depends on the values of some vm subsystem attributes. Figure 12–6 shows some of the attributes that control paging and swapping.

Figure 12–6: Paging and Swapping Attributes



Detailed descriptions of the attributes are as follows:

- `vm_page_free_target` — Specifies a threshold value that stops paging. When the number of pages on the free page list reaches this value, paging stops. The default value of the `vm_page_free_target` attribute is based on the amount of memory in the system. Use Table 12–1 to determine the default value for your system.

Table 12–1: Default Values for vm_page_free_target Attribute

Size of Memory	Value of vm_page_free_target
Up to 512 MB	128
513 MB to 1024 MB	256
1025 MB to 2048 MB	512
2049 MB to 4096 MB	768
More than 4096 MB	1024

- `vm_page_free_min` — Specifies a threshold value at which a page must be reclaimed for each page allocated. The default value is twice the value of the `vm_page_free_reserved` attribute.

- vm_page_free_reserved — Specifies a threshold value that determines when memory is limited to privileged tasks. When the number of pages on the free page list falls below this value, only privileged tasks can get memory. The default value is 10 pages.
- vm_page_free_swap — Specifies a threshold value that begins swapping idle tasks. When the number of pages on the free page list falls below this value, idle task swapping begins. The default value is calculated using this formula:

  ```
  vm_page_free_min + ((vm_page_free_target - vm_page_free_min) / 2)
  ```

- vm_page_free_optimal — Specifies a threshold value that begins hard swapping. When the number of pages on the free list falls below this value for five seconds, hard swapping begins. The first processes to be swapped out include those with the lowest scheduling priority and those with the largest resident set size. The default value is calculated using this formula:

  ```
  vm_page_free_min + ((vm_page_free_target - vm_page_free_min) / 2)
  ```

- vm_page_free_hardswap — Specifies a threshold value that stops page swapping. When the number of pages on the free list reaches this value, paging stops. The default value is the value of the vm_page_free_target attribute multiplied by 16.

See Section 12.5 for information about modifying paging and swapping attributes.

The following sections describe the page reclamation procedure in detail.

12.1.4.1 Modified Page Prewriting

The virtual memory subsystem attempts to prevent memory shortages by prewriting modified inactive and UBC LRU pages to disk. To reclaim a page that has been prewritten, the virtual memory subsystem only needs to validate the page, which can improve performance. See Section 12.1.1 for information about page lists.

When the virtual memory subsystem anticipates that the pages on the free list will soon be depleted, it prewrites to disk the oldest modified (dirty) pages that are currently being used by processes or the UBC.

The value of the vm subsystem attribute vm_page_prewrite_target determines the number of inactive pages that the subsystem will prewrite and keep clean. The default value is vm_page_free_target * 2.

The vm_ubcdirtypercent attribute specifies the modified UBC LRU page threshold. When the number of modified UBC LRU pages is more than this value, the virtual memory subsystem prewrites to disk the oldest modified

UBC LRU pages. The default value of the vm_ubcdirtypercent attribute is 10 percent of the total UBC LRU pages.

In addition, the sync function periodically flushes (writes to disk) system metadata and data from all unwritten memory buffers. For example, the data that is flushed includes, for UFS, modified inodes and delayed block I/O. Commands, such as the shutdown command, also issue their own sync functions. To minimize the impact of I/O spikes caused by the sync function, the value of the vm subsystem attribute ubc_maxdirtywrites specifies the maximum number of disk writes that the kernel can perform each second. The default value is five I/O operations per second.

12.1.4.2 Reclaiming Memory by Paging

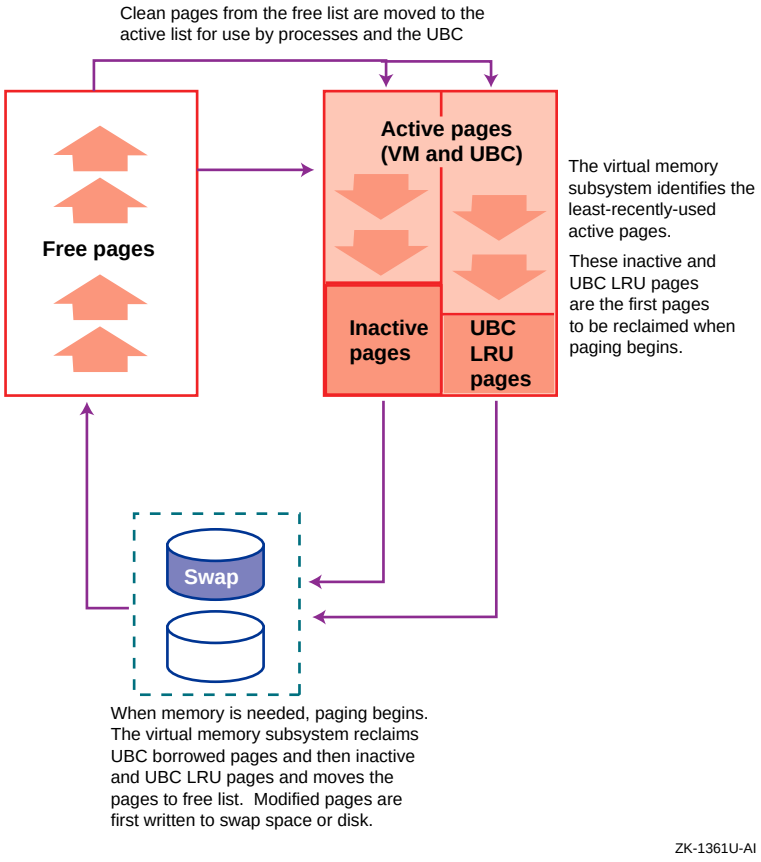
When the memory demand is high and the number of pages on the free page list falls below the value of the vm subsystem attribute vm_page_free_target, the virtual memory subsystem uses paging to replenish the free page list. The page-out daemon and task swapper daemon are extensions of the page reclamation code, which controls paging and swapping.

The paging process is as follows:

1. The page reclamation code activates the page-stealer daemon, which first reclaims the clean pages that the UBC has borrowed from the virtual memory subsystem, until the size of the UBC reaches the borrowing threshold that is specified by the value of the ubc_borrowpercent attribute (the default is 20 percent). Freeing borrowed UBC pages is a fast way to reclaim pages, because UBC pages are usually not modified. If the reclaimed pages are dirty (modified), their contents must be written to disk before the pages can be moved to the free page list.
2. If freeing clean UBC borrowed memory does not sufficiently replenish the free list, a **page out** occurs. The page-stealer daemon reclaims the oldest inactive and UBC LRU pages from the active page list, moves the contents of the modified pages to disk, and puts the clean pages on the free list.
3. Paging becomes increasingly aggressive if the number of free pages continues to decrease. If the number of pages on the free page list falls below the value of the vm subsystem attribute vm_page_free_min (the default is 20 pages), a page must be reclaimed for each page taken from the list.

Figure 12–7 shows the movement of pages during paging operations.

Figure 12–7: Paging Operation



Paging stops when the number of pages on the free list increases to the limit specified by the `vm` subsystem attribute `vm_page_free_target`. However, if paging individual pages does not sufficiently replenish the free list, swapping is used to free a large amount of memory (see Section 12.1.4.3).

12.1.4.3 Reclaiming Memory by Swapping

If there is a continuously high demand for memory, the virtual memory subsystem may be unable to replenish the free page list by reclaiming single pages. To dramatically increase the number of clean pages, the virtual memory subsystem uses swapping to suspend processes, which reduces the demand for physical memory.

The task swapper will **swap out** a process by suspending the process, writing its resident set to swap space, and moving the clean pages to the free page list. Swapping has a serious impact on system performance because a

swapped out process cannot execute, and should be avoided on VLM systems and systems running large programs.

The point at which swapping begins and ends is controlled by a number of `vm` subsystem attributes, as follows:

- Idle task swapping begins when the number of pages on the free list falls below the value of the `vm_page_free_swap` attribute for a period of time. The task swapper suspends all tasks that have been idle for 30 seconds or more.
- Hard task swapping begins when the number of pages on the free page list falls below the value of the `vm_page_free_optimal` attribute for more than five seconds. The task swapper suspends, one at a time, the tasks with the lowest priority and the largest resident set size.
- Swapping stops when the number of pages on the free list increases to the value of the `vm_page_free_hardswap` attribute.
- A **swap in** occurs when the number of pages on the free list increases to the value of the `vm_page_free_optimal` attribute for a period of time. The value of the `vm_inswappedmin` attribute specifies the minimum amount of time, in seconds, that a task must remain in the inswapped state before it can be moved out of swap space. The default value is 1 second. The task's working set is then paged in from swap space, and the task can now execute. You can modify the value of the `vm_inswappedmin` attribute without rebooting the system.

You may be able to improve system performance by modifying the attributes that control when swapping begins and ends, as described in Section 12.5. Large-memory systems or systems running large programs should avoid paging and swapping, if possible.

Increasing the rate of swapping (swapping earlier during page reclamation) may increase throughput. As more processes are swapped out, fewer processes are actually executing and more work is done. Although increasing the rate of swapping moves long-sleeping threads out of memory and frees memory, it may degrade interactive response time because when an outswapped process is needed, it will have a long latency period.

Decreasing the rate of swapping (by swapping later during page reclamation) may improve interactive response time, but at the cost of throughput. See Section 12.5.2 for more information about changing the rate of swapping.

To facilitate the movement of data between memory and disk, the virtual memory subsystem uses synchronous and asynchronous swap buffers. The virtual memory subsystem uses these two types of buffers to immediately satisfy a page-in request without having to wait for the completion of a page-out request, which is a relatively slow process.

Synchronous swap buffers are used for page-in page faults and for swap outs. Asynchronous swap buffers are used for asynchronous page outs and for prewriting modified pages. See Section 12.5.7 for swap buffer tuning information.

12.2 Configuring Swap Space for High Performance

Use the `swapon` command to display swap space, and to configure additional swap space after system installation. To make this additional swap space permanent, use the `vm` subsystem attribute `swapdevice` to specify swap devices in the `/etc/sysconfigtab` file. For example:

```
vm:
      swapdevice=/dev/disk/dsk0b,/dev/disk/dsk0d
```

See Chapter 3 for information about modifying kernel subsystem attributes.

See Section 4.4.1.8 or Section 12.1.3 for information about swap space allocation modes and swap space requirements.

The following list describes how to configure swap space for high performance:

- Ensure that all your swap devices are configured when you boot the system, instead of adding swap space while the system is running.
- Use fast disks for swap space to decrease page-fault latency.
- Use disks that are not busy for swap space.
- Spread out swap space across multiple disks; do not put multiple swap partitions on the same disk. This makes paging and swapping more efficient and helps to prevent any single adapter, disk, or bus from becoming a bottleneck. The page reclamation code uses a form of disk striping (known as **swap-space interleaving**) that improves performance when data is written to multiple disks.
- Spread out your swap disks across multiple I/O buses to prevent a single bus from becoming a bottleneck.
- Configure multiple swap devices as individual devices (or LSM volumes) instead of striping the devices and configuring only one logical swap device.
- If you are paging heavily and cannot increase the amount of memory in your system, consider using RAID5 for swap devices. See Chapter 9 for more information about RAID5.

See the *System Administration* manual for more information about adding swap devices. See Chapter 9 for more information about configuring and tuning disks for high performance and availability.

12.3 Monitoring Memory Statistics

Table 12–2 describes the tools that you can use to display memory usage information.

Table 12–2: Tools to Display Virtual Memory and UBC

Tools	Description	Reference
`vmstat`	Displays information about process threads, virtual memory usage (page lists, page faults, page ins, and page outs), interrupts, and CPU usage (percentages of user, system and idle times).	Section 12.3.1
`ps`	Displays current statistics for running processes, including CPU usage, the processor and processor set, and the scheduling priority. The `ps` command also displays virtual memory statistics for a process, including the number of page faults, page reclamations, and page ins; the percentage of real memory (resident set) usage; the resident set size; and the virtual address size.	Section 12.3.2)
`swapon`	Displays information about swap space utilization and the total amount of allocated swap space, swap space in use, and free swap space for each swap device. You can also use this command to allocate additional swap space.	Section 12.3.3
`(dbx) print ufs_geta-page_stats`	Reports UBC statistics and examines the `ufs_getapage_stats` data structure, which contains information about UBC page usage.	Section 12.3.4
`sys_check`	Analyzes system configuration and displays statistics, providing warnings and tuning guidelines if necessary.	Section 2.3.3

Table 12–2: Tools to Display Virtual Memory and UBC (cont.)

Tools	Description	Reference
`uerf -r 300`	Displays total system memory.	See `uerf`(8) for more information
`ipcs`	Displays interprocess communication (IPC) statistics for currently active message queues, shared-memory segments, semaphores, remote queues, and local queue headers. The information provided in the following fields reported by the `ipcs –a` command can be especially useful: `QNUM`, `CBYTES`, `QBYTES`, `SEGSZ`, and `NSEMS`.	See `ipcs`(1) for more information

The following sections describe the `vmstat`, `ps`, `swapon`,and `dbx` tools in detail.

12.3.1 Displaying Memory by Using the vmstat Command

To display the virtual memory, process, and CPU statistics, enter:

```
# /usr/ucb/vmstat
```

Information similar to the following is displayed:

```
Virtual Memory Statistics: (pagesize = 8192)
procs        memory            pages                          intr        cpu
r  w  u  act  free wire  fault cow zero react pin pout   in  sy  cs  us sy  id
2 66 25  6417 3497 1570  155K  38K  50K    0  46K    0    4 290 165   0  2  98
4 65 24  6421 3493 1570   120    9   81    0    8    0  585 865 335  37 16  48
2 66 25  6421 3493 1570    69    0   69    0    0    0  570 968 368   8 22  69
4 65 24  6421 3493 1570    69    0   69    0    0    0  554 768 370   2 14  84
4 65 24  6421 3493 1570    69    0   69    0    0    0  865  1K 404   4 20  76
 1           2           3                              4          5
```

The first line of the `vmstat` output is for all time since a reboot, and each subsequent report is for the last interval.

The `vmstat` command includes information that you can use to diagnose CPU and virtual memory problems. Examine the following fields:

1 Process information (`procs`):

- `r` — Number of threads that are running or can run.
- `w` — Number of threads that are waiting interruptibly (waiting for an event or a resource, but can be interrupted or suspended). For example, the thread can accept user signals or be swapped out of memory.

- `u` — Number of threads that are waiting uninterruptibly (waiting for an event or a resource, but cannot be interrupted or suspended). For example, the thread cannot accept user signals; it must come out of the wait state to take a signal. Processes that are waiting uninterruptibly cannot be stopped by the `kill` command.

2 Virtual memory information (`memory`):

- `act` — Number of pages on the active list, including inactive pages and UBC LRU pages.
- `free` — Number of pages on the free list.
- `wire` — Number of pages on the wired list. Pages on the wired list cannot be reclaimed.

See Section 12.1.1 for more information on page lists.

3 Paging information (`pages`):

- `fault` — Number of address translation faults.
- `cow` — Number of copy-on-write page faults. These page faults occur if the requested page is shared by a parent process and a child process, and one of these processes needs to modify the page. If a copy-on-write page fault occurs, the virtual memory subsystem loads a new address into the translation buffer, and then copies the contents of the requested page into this address, so that the process can modify it.
- `zero` — Number of zero-filled-on-demand page faults. These page faults occur if a requested page is not located in the internal data structures and has never been referenced. If a zero-filled-on-demand page fault occurs, the virtual memory subsystem allocates an available page of physical memory, fills the page with zeros, and then enters the address into the page table.
- `react` — Number of pages that have been faulted (touched) while on the inactive page list.
- `pin` — Number of requests for pages from the page-stealer daemon.
- `pout` — Number of pages that have been paged out to disk.

4 Interrupt information (`intr`):

- `in` — Number of nonclock device interrupts per second.
- `sy` — Number of system calls called per second.
- `cs` — Number of task and thread context switches per second.

5 CPU usage information (`cpu`):

- us — Percentage of user time for normal and priority processes. User time includes the time the CPU spent executing library routines.
- sy — Percentage of system time. System time includes the time the CPU spent executing system calls.
- id — Percentage of idle time.

See Section 12.3.1 for information about using the vmstat command to monitor CPU usage.

To use the vmstat command to diagnose a memory performance problem:

- Check the size of the free page list (free). Compare the number of free pages to the values for the active pages (act) and the wired pages (wire). The sum of the free, active, and wired pages should be close to the amount of physical memory in your system. Although the value for free should be small, if the value is consistently small (less than 128 pages) and accompanied by excessive paging and swapping, you may not have enough physical memory for your workload.
- Examine the pout field. If the number of page outs is consistently high, you may have insufficient memory.
- The following command output may indicate that the size of the UBC is too small for your configuration:
  - The output of the vmstat or monitor command shows excessive file system page-in activity, but little or no page-out activity or shows a very low free page count.
  - The output of the iostat command shows little or no swap disk I/O activity or shows excessive file system I/O activity. See Section 9.2 for more information.

Excessive paging also can increase the miss rate for the secondary cache, and may be indicated by the following output:

- The output of the ps command shows high task swapping activity. See Section 12.3.2 for more information.
- The output of the swapon command shows excessive use of swap space. See Section 12.3.3 for more information.

To display statistics about physical memory use, enter:

```
# vmstat -P
```

Information similar to the following is displayed:

```
Total Physical Memory =   512.00 M
                      =    65536 pages
Physical Memory Clusters:
```

```
 start_pfn     end_pfn        type  size_pages / size_bytes
         0         256         pal         256 /    2.00M
       256       65527          os       65271 /  509.93M
     65527       65536         pal           9 /   72.00k

Physical Memory Use:

 start_pfn     end_pfn        type  size_pages / size_bytes
       256         280   unixtable          24 /  192.00k
       280         287    scavenge           7 /   56.00k
       287         918        text         631 /    4.93M
       918        1046        data         128 /    1.00M
      1046        1209         bss         163 /    1.27M
      1210        1384      kdebug         174 /    1.36M
      1384        1390     cfgmgmt           6 /   48.00k
      1390        1392       locks           2 /   16.00k
      1392        1949   unixtable         557 /    4.35M
      1949        1962        pmap          13 /  104.00k
      1962        2972    vmtables        1010 /    7.89M
      2972       65527     managed       62555 /  488.71M
                           ============================
         Total Physical Memory Use:      65270 /  509.92M

Managed Pages Break Down:

       free pages = 1207
     active pages = 25817
   inactive pages = 20103
      wired pages = 15434
        ubc pages = 15992
        ==================
            Total = 78553

WIRED Pages Break Down:

   vm wired pages = 1448
  ubc wired pages = 4550
  meta data pages = 1958
     malloc pages = 5469
     contig pages = 159
    user ptepages = 1774
  kernel ptepages = 67
    free ptepages = 9
        ==================
            Total = 15434
```

See `vmstat`(1) for more information about this command and its options. See Section 12.4 for information about increasing memory resources.

12.3.2 Displaying Memory by Using the ps Command

To display the current state of the system processes and how they use memory, enter:

```
# /usr/ucb/ps aux
```

Information similar to the following is displayed:

```
USER  PID  %CPU %MEM   VSZ   RSS  TTY S    STARTED      TIME  COMMAND
chen  2225  5.0  0.3  1.35M  256K p9  U    13:24:58  0:00.36  cp /vmunix /tmp
root  2236  3.0  0.5  1.59M  456K p9  R  + 13:33:21  0:00.08  ps aux
sorn  2226  1.0  0.6  2.75M  552K p9  S  + 13:25:01  0:00.05  vi met.ps
```

```
root   347  1.0  4.0  9.58M  3.72 ??  S      Nov 07 01:26:44  /usr/bin/X11/X -a
root  1905  1.0  1.1  6.10M  1.01 ??  R    16:55:16  0:24.79  /usr/bin/X11/dxpa
mat   2228  0.0  0.5  1.82M  504K p5  S  + 13:25:03  0:00.02  more
mat   2202  0.0  0.5  2.03M  456K p5  S    13:14:14  0:00.23  -csh (csh)
root     0  0.0 12.7   356M  11.9 ??  R <  Nov 07 3-17:26:13  [kernel idle]
           1   2    3   4       5               6       7
```

The ps command displays a snapshot of system processes in order of decreasing CPU usage, including the execution of the ps command itself. By the time the ps command executes, the state of system processes has probably changed.

The ps command output includes the following information that you can use to diagnose CPU and virtual memory problems:

1 Percentage of CPU time usage (%CPU).

2 Percentage of real memory usage (%MEM).

3 Process virtual address size (VSZ) — This is the total amount of anonymous memory allocated to the process (in bytes).

4 Real memory (resident set) size of the process (RSS) — This is the total amount of physical memory, in bytes, mapped to virtual pages (that is, the total amount of memory that the application has physically used). Shared memory is included in the resident set size figures; as a result, the total of these figures may exceed the total amount of physical memory available on the system.

5 Process status or state (S) — This specifies whether a process is in one of the following states:

- Runnable (R)
- Sleeping (S) — Process has been waiting for an event or a resource for less than 20 seconds, but it can be interrupted or suspended. For example, the process can accept user signals or be swapped out.
- Uninterruptible sleeping (U) — Process is waiting for an event or a resource, but cannot be interrupted or suspended. You cannot use the kill command to stop these processes; they must come out of the wait state to accept the signal.
- Idle (I) — Process has been sleeping for more than 20 seconds.
- Stopped (T) — Process has been stopped.
- Halted (H) — Process has been halted.
- Swapped out (W) — Process has been swapped out of memory.
- Locked into memory (L) — Process has been locked into memory and cannot be swapped out.
- Has exceeded the soft limit on memory requirements (>).

- A process group leader with a controlling terminal (+).
- Has a reduced priority (N).
- Has a raised priority (<).

6 Current CPU time used (TIME), in the format *hh*:*mm*:*ss*.*ms*.

7 The command that is running (COMMAND).

From the output of the ps command, you can determine which processes are consuming most of your system's CPU time and memory resources, and whether processes are swapped out. Concentrate on processes that are running or paging. Here are some concerns to keep in mind:

- If a process is using a large amount of memory (see the RSS and VSZ fields), the process may have excessive memory requirements. See Section 7.1 for information about decreasing an application's use of memory.
- If duplicate processes are running, use the kill command to terminate duplicate processes. See kill(1) for more information.
- If a process is using a large amount of CPU time, it may be in an infinite loop. You may have to use the kill command to terminate the process and then correct the problem by making changes to its source code.

  You can also use the Class Scheduler to allocate a percentage of CPU time to a specific task or application (see Section 13.2.2) or lower the process' priority by using either the nice or renice command. These commands have no effect on memory usage by a process. See nice(1) or renice(8) for more information.
- Check the processes that are swapped out. Examine the S (state) field. A W entry indicates a process that has been swapped out. If processes are continually being swapped out, this could indicate a lack of memory resources. See Section 12.4 for information about increasing memory resources.

See ps(1) for more information about this command and its options.

12.3.3 Displaying Swap Space Usage by Using the swapon Command

To display information about your swap device configuration, including the total amount of allocated swap space, the amount of swap space that is being used, and the amount of free swap space, enter:

```
# /usr/sbin/swapon -s
```

Infomation for each swap partition is displayed similar to the following:

```
Swap partition /dev/disk/dsk1b (default swap):
    Allocated space:        16384 pages (128MB)
    In-use space:           10452 pages ( 63%)
```

```
    Free space:              5932 pages ( 36%)

Swap partition /dev/disk/dsk4c:
    Allocated space:        128178 pages (1001MB)
    In-use space:            10242 pages (  7%)
    Free space:             117936 pages ( 92%)

Total swap allocation:

    Allocated space:        144562 pages (1.10GB)
    Reserved space:          34253 pages ( 23%)
    In-use space:            20694 pages ( 14%)
    Available space:        110309 pages ( 76%)
```

You can configure swap space when you first install the operating system, or you can add swap space at a later date. Application messages, such as the following, usually indicate that not enough swap space is configured into the system or that a process limit has been reached:

```
"unable to obtain requested swap space"
"swap space below 10 percent free"
```

See Section 4.4.1.8 or Section 12.1.3 for information about swap space requirements. See Section 12.2 for information about adding swap space and distributing swap space for high performance.

See swapon(2) for more information about this command and its options.

12.3.4 Displaying the UBC by Using the dbx Debugger

If you have not disabled read-ahead, you can display the UBC by using the `dbx print` command to examine the `ufs_getapage_stats` data structure. For example:

```
# /usr/ucb/dbx -k /vmunix /dev/mem  (dbx) print ufs_getapage_stats
```

Information similar to the following is displayed:

```
struct {
    read_looks = 2059022
    read_hits = 2022488
    read_miss = 36506
    alloc_error = 0
    alloc_in_cache = 0
}
(dbx)
```

To calculate the hit rate, divide the value of the `read_hits` field by the value of the `read_looks` field. A good hit rate is a rate above 95 percent. In the previous example, the hit rate is approximately 98 percent.

See dbx(1) for more information about this command and its options.

12.4 Tuning to Provide More Memory to Processes

If your system is paging or swapping, you may be able to increase the memory that is available to processes by tuning various kernel subsystem attributes.

Table 12–3 shows the guidelines for increasing memory resources to processes and lists the performance benefits as well as trade offs. Some of the guidelines for increasing the memory available to processes may affect UBC operation and file-system caching. Adding physical memory to your system is the best way to stop paging or swapping.

Table 12–3: Memory Resource Tuning Guidelines

Performance Benefit	Guideline	Tradeoff
Decrease CPU load and demand for memory	Reduce the number of processes running at the same time (Section 12.4.1)	System performs less work
Free memory	Reduce the static size of the kernel (Section 12.4.2)	Not all functionality may be available
Improve network throughput under a heavy load	Increase the percentage of memory reserved for kernel `malloc` allocations (Section 12.4.3)	Consumes memory
Improve system response time when memory is low	Decrease cache sizes (Section 11.1)	May degrade file-system performance
Free memory	Reduce process memory requirements (Section 7.1.6)	Program may not run optimally

The following sections discuss these tuning guidelines in more detail.

12.4.1 Reducing the Number of Processes Running Simultaneously

You can improve performance and reduce the demand for memory by running fewer applications simultaneously. Use the `at` or `batch` command to run applications at offpeak hours.

See `at`(1) for more information.

12.4.2 Reducing the Static Size of the Kernel

You can reduce the static size of the kernel by deconfiguring any unnecessary subsystems. Use the `sysconfig` command to display the configured subsystems and to delete subsystems. Be sure not to remove any subsystems or functionality that is vital to your environment.

See Chapter 3 for information about modifying kernel subsystem attributes.

12.4.3 Increasing the Memory Reserved for Kernel malloc Allocations

If you are running a large Internet application, you may need to increase the amount of memory reserved for the kernel `malloc` subsystem. This improves network throughput by reducing the number of packets that are dropped while the system is under a heavy network load. However, increasing this value consumes memory.

Related Attribute

The following list describes the `generic` subsystem attribute that relates to the memory reserved for kernel allocations:

- `kmemreserve_percent` — Specifies the percentage of physical memory reserved for kernel memory allocations that are less than or equal to the page size (8 KB).

  Value: 1 to 75
  Default: 0, which actually specifies 0.4 percent of available memory or 256 KB, whichever is smaller.

  You can modify the `kmemreserve_percent` attribute without rebooting.

When to Tune

You might want to increase the value of the `kmemreserve_percent` attribute if the output of the `netstat -d -i` command shows dropped packets, or if the output of the `vmstat -M` command shows dropped packets under the `fail_nowait` heading. This may occur under a heavy network load.

See Chapter 3 for information about modifying kernel subsystem attributes.

12.5 Modifying Paging and Swapping Operations

You might improve performance by modifying paging and swapping operations that are described in the following sections:

- Increasing the paging threshold (Section 12.5.1)
- Managing the rate of swapping (Section 12.5.2)
- Enabling aggressive swapping (Section 12.5.3)
- Limiting process resident set size (Section 12.5.4)
- Managing the rate of dirty page prewriting (Section 12.5.5)
- Managing page-in and page-out cluster sizes (Section 12.5.6)
- Managing I/O requests (Section 12.5.7)

- Using memory locking (Section 7.1.7)

12.5.1 Increasing the Paging Threshold

Paging is the transfer of program segments (pages) into and out of memory. Excessive paging is not desired. You can specify the number of pages on the free list before paging begins. See Section 12.1.4 for more information on paging.

Related Attribute

The vm subsystem vm_page_free_target specifies the minimum number of pages on the free list before paging begins. The default value of the vm_page_free_target attribute is based on the amount of memory in the system.

Use the following table to determine the default value for your system:

Size of Memory	Value of vm_page_free_target
Up to 512 MB	128
513 MB to 1024 MB	256
1025 MB to 2048 MB	512
2049 MB to 4096 MB	768
More than 4096 MB	1024

You can modify the vm_page_free_target attribute without rebooting the system.

When to Tune

Do not decrease the value of the vm_page_free_target attribute.

Do not increase the value of the vm_page_free_target attribute if the system is not paging. You might want to increase the value of the vm_page_free_target attribute if you have sufficient memory resources, and your system experiences performance problems when a severe memory shortage occurs. However, increasing this value might increase paging activity on a low-memory system and can waste memory if it is set too high. See Section 12.1.4 for information about paging and swapping attributes.

If you increase the default value of the vm_page_free_target attribute, you may also want to increase the value of the vm_page_free_min attribute.

See Chapter 3 for information about modifying kernel subsystem attributes.

12.5.2 Managing the Rate of Swapping

Swapping begins when the free page list falls below the swapping threshold. Excessive swapping is not desired. You can specify when swapping begins and ends. See Section 12.1.4 for more information on swapping.

Related Attributes

The following list describes the vm subsystem attributes that relate to modified page prewriting:

- vm_page_free_optimal — Specifies a threshold value that begins hard swapping. When the number of pages on the free list falls below this value for five seconds, hard swapping begins.

  Value: 0 to 2,147,483,647
  Default value: Automatically scaled by using this formula:
  vm_page_free_min + ((vm_page_free_target - vm_page_free_min)/ 2)

- vm_page_free_min — Specifies a threshold value that starts page swapping. When the number of pages on the free page list falls below this value, paging starts.

  Value: 0 to 2,147,483,647
  Default value: 20 (pages, or twice the amount of vm_page_free_reserved)

- vm_page_free_reserved — Specifies a threshold value that determines when memory is limited to privileged tasks. When the number of pages on the free page list falls below this value, only privileged tasks can get memory.

  Value: 1 to 2,147,483,647
  Default value: 10 (pages)

- vm_page_free_target — Specifies that when the number pages on the free page list reaches this value, paging stops.

  The default value is based on the amount of managed memory that is available on the system, as follows:

Available Memory (MB)	vm_page_free_target (pages)
Less than 512	128
512 to 1023	256
1024 to 2047	512

Available Memory (MB)	vm_page_free_target (pages)
2048 to 4095	768
4096 and higher	1024

You can modify the vm_page_free_optimal, vm_page_free_min, and vm_page_free_target attributes without rebooting the system. See Chapter 3 for information about modifying kernel subsystem attributes.

When to Tune

Do not change the value of the vm_page_free_optimal attribute if the system is not paging.

Decreasing the value of the vm_page_free_optimal attribute improves interactive response time, but decreases throughput.

Increasing the value of the vm_page_free_optimal attribute moves long-sleeping threads out of memory, frees memory, and increases throughput. As more processes are swapped out, fewer processes are actually executing and more work is done. However, when an outswapped process is needed, it will have a long latency and might degrade interactive response time.

Increase the value of the vm_page_free_optimal only by two pages at a time. Do not specify a value that is more than the value of the vm subsystem attribute vm_page_free_target.

12.5.3 Enabling Aggressive Task Swapping

Swapping begins when the free page list falls below the swapping threshold, as specified by the vm subsystem vm_page_free_swap attribute. Excessive swapping is not desired. You can specify whether or not idle tasks are aggressively swapped out. See Section 12.1.4 for more information on swapping.

Related Attribute

The vm subsystem vm_aggressive_swap specifies whether or not the task swapper aggressively swaps out idle tasks.

Value: 1 or 0
Default value: 0 (disabled)

When to Tune

Aggressive task swapping improves system throughput, but it degrades the interactive response performance. Usually, you do not need to enable aggressive task swapping.

You can modify the vm_aggressive_swap attribute without rebooting. See Chapter 3 for information about modifying kernel attributes.

12.5.4 Limiting the Resident Set Size to Avoid Swapping

By default, Tru64 UNIX does not limit the resident set size for a process. Applications can set a process-specific limit on the number of pages resident in memory by specifying the RLIMIT_RSS resource value in a setrlimit() call. However, applications are not required to limit the resident set size of a process and there is no systemwide default limit. Therefore, the resident set size for a process is limited only by system memory restrictions. If the demand for memory exceeds the number of free pages, processes with large resident set sizes are likely candidates for swapping. See Section 12.1.4 for more information on swapping.

To avoid swapping a process because it has a large resident set size, you can specify process-specific and systemwide limits for resident set sizes.

Related Attributes

The following list describes the vm subsystem attributes that relate to limiting the resident set size:

- anon_rss_enforce — Specifies different levels of control over process set sizes and when the pages that a process is using in anonymous memory are swapped out (blocking the process) during times of contention for free pages.

  Value: no limit (0), a soft limit (1), or a hard limit (2)
  Default value: 0 (no limit)

  Setting anon_rss_enforce to either 1 or 2 allows you to enforce a systemwide limit on resident set size for a process through the vm_rss_max_percent attribute.

  Setting anon_rss_enforce to 1 (a soft limit) enables finer control over process blocking and paging of anonymous memory by allowing you to set the vm_rss_block_target and vm_rss_wakeup_target attributes.

- vm_rss_max_percent — Specifies a percentage of the total pages of anonymous memory on the system that is the systemwide limit on the resident set size for any process. The value of this attribute has an effect only if the anon_rss_enforce attribute is set to 1 or 2.

  Value: 0 to 100
  Default value: 100 percent

  You can decrease this percentage to enforce a systemwide limit on the resident set size for any process. Be aware, however, that this limit applies to privileged and unprivileged processes and will override a

larger resident set size that may be specified for a process through the `setrlimit()` call.

- `vm_rss_block_target` — Specifies a threshold number of free pages that will start swapping anonymous memory from the resident set of a process. Paging of anonymous memory starts when the number of free pages meets or exceeds this value. The process is blocked until the number of free pages reaches the value set by the `vm_rss_wakeup_target` attribute.

  Value: 0 to 2,147,483,647
  Default value: Same as `vm_page_free_optimal`

  Increasing the value starts paging of anonymous memory earlier than when hard swapping occurs. Decreasing the value delays paging of anonymous memory beyond the point at which hard swapping occurs.

- `vm_rss_wakeup_target` — Specifies a threshold number of free pages that will unblock a process whose anonymous memory is swapped out. The process is unblocked when the number of free pages meets this value.

  Value: 0 to 2,147,483,647
  Default value: Same as `vm_page_free_optimal`

  Increasing the value frees more memory before unblocking the task. Decreasing the value unblocks tasks sooner, but less memory is freed.

- `vm_page_free_optimal` — Specifies a threshold value that begins hard swapping. When the number of pages on the free list falls below this value for five seconds, hard swapping begins.

  Value: 0 to 2,147,483,647
  Default value: Automatically scaled by using this formula: `vm_page_free_min` + ((`vm_page_free_target` - `vm_page_free_min`) / 2)

When to Tune

You do not need to limit resident set sizes if the system is not paging.

If you limit the resident set size, either for a specific process or systemwide, you must also use the `vm` subsystem attribute `anon_rss_enforce` to set either a soft or hard limit on the size of a resident set.

If you enable a hard limit, a task's resident set cannot exceed the limit. If a task reaches the hard limit, pages of the task's anonymous memory are moved to swap space to keep the resident set size within the limit.

If you enable a soft limit, anonymous memory paging will start when the following conditions are met:

- A task's resident set exceeds the systemwide or per-process limit.
- The number of pages of the free page list is less than the value of the vm_rss_block_target attribute.

You cannot modify the anon_rss_enforce attribute without rebooting the system. You can modify the vm_page_free_optimal, vm_rss_maxpercent, vm_rss_block_target, and vm_rss_wakeup_target attributes without rebooting the system.

12.5.5 Managing Modified Page Prewriting

The vm subsystem attempts to prevent a memory shortage by prewriting modified (dirty) pages to disk. To reclaim a page that was prewritten, the virtual memory subsystem only needs to validate the page, which can improve performance. When the virtual memory subsystem anticipates that the pages on the free list will soon be depleted, it prewrites to disk the oldest inactive and UBC LRU pages. You can tune attributes that relate to prewriting. See Section 12.1.4.1 for more information about prewriting.

Related Attributes

The following list describes the vm subsystem attributes that relate to modified page prewriting:

- vm_ubcdirtypercent — Specifies the percentage of pages that must be dirty (modified) before the UBC starts writing them to disk.

  Value: 0 to 100
  Default: 10 percent

- vm_page_prewrite_target — Specifies the maximum number of modified UBC (LRU) pages that the vm subsystem will prewrite to disk if it anticipates running out of memory.

  Value: 0 to 2,147,483,647
  Default: vm_page_free_target * 2

- vm_page_free_target — Specifies that when the number pages on the free page list reaches this value, paging stops.

  The following table shows the default value is based on the amount of managed memory that is available on the system:

Available Memory (MB)	vm_page_free_target (pages)
Less than 512	128
512 to 1023	256
1024 to 2047	512

Available Memory (MB)	vm_page_free_target (pages)
2048 to 4095	768
4096 and higher	1024

You can modify the vm_page_prewrite_target or vm_ubcdirtypercent attribute without rebooting the system.

When to Tune

You do not need to modify the value of the vm_page_prewrite_target attribute if the system is not paging.

Decreasing the value of the vm_page_prewrite_target attribute will improve peak workload performance, but it will cause a drastic performance degradation when memory is exhausted.

Increasing the value of the vm_page_prewrite_target attribute will:

- Prevent a drastic performance degradation when memory is exhausted, but will also reduce peak workload performance.
- Increase the amount of continuous disk I/O, but provide better file system integrity if a system crash occurs.

Increase the value of the vm_page_prewrite_target attribute by increments of 64 pages.

To increase the rate of UBC LRU dirty page prewriting, decrease the value of the vm_ubcdirtypercent attribute by increments of 1 percent.

See Chapter 3 for information about modifying kernel attributes.

12.5.6 Managing Page-In and Page-Out Clusters Sizes

The virtual memory subsystem reads in and writes out additional pages to the swap device in an attempt to anticipate the number of pages that it will need. You can specify the number of additional pages to the swap device.

Related Attributes

The following list describes the vm subsystem attributes that relate to reading and writing pages:

- vm_max_rdpgio_kluster — Specifies the size, in bytes, of the largest page-in (read) cluster that is passed to the swap device.

  Value: 8192 to 131,072
  Default: 16,384 (bytes) (16 KB)

- vm_max_wrpgio_kluster — Specifies the size, in bytes, of the largest page-out (write) cluster that is passed to the swap device.

  Value: 8192 to 131,072
  Default: 32,768 (bytes) (32 KB)

You cannot modify the vm_max_rdpgio_kluster and vm_max_wrpgio_kluster attributes without rebooting the system. See Chapter 3 for information about modifying kernel subsystem attributes.

When to Tune

You might want to increase the value of the vm_max_rdpgio_kluster attribute if you have a large-memory system and you are swapping processes. Increasing the value increases the peak workload performance because more pages will be in memory and the system will spend less time page faulting, but will consume more memory and decrease system performance.

You may want to increase the value of the vm_max_wrpgio_kluster attribute if you are paging and swapping processes. Increasing the value improves the peak workload performance and conserves memory, but might cause more page ins and decrease the total system workload performance.

12.5.7 Managing I/O Requests on the Swap Partition

Swapping begins when the free page list falls below the swapping threshold. Excessive swapping is not desired. You can specify the number of outstanding synchronous and asynchronous I/O requests that can be on swap partitions at one time. See Section 12.1.4 for more information on swapping.

Synchronous swap requests are used for page-in operations and task swapping. Asynchronous swap requests are used for page-out operations and for prewriting modified pages.

Related Attributes

The following list describes the vm subsystem attributes that relate to requests in swap partitions:

- vm_syncswapbuffers — Specifies the number of synchronous I/O requests that can be outstanding to the swap partitions at one time. Synchronous swap requests are used for page-in operations and task swapping.

  Value: 1 to 2,147,483,647
  Default: 128 (requests)

- vm_asyncswapbuffers — Specifies the number of asynchronous I/O requests per swap partition that can be outstanding at one time.

Asynchronous swap requests are used for page-out operations and for prewriting modified pages.

Value: 0 to 2,147,483,647
Default: 4 (requests)

When to Tune

The value of the vm_syncswapbuffers attribute should be equal to the approximate number of simultaneously running processes that the system can easily support. Increasing the value increases overall system throughput, but it consumes memory.

The value of the vm_asyncswapbuffers attribute should be equal to the approximate number of I/O transfers that a swap device can support at one time. If you are using LSM, you might want to increase the value of the vm_asyncswapbuffers attribute, which causes page-in requests to lag asynchronous page-out requests. Decreasing the value will use more memory, but it will improve the interactive response time.

You can modify the vm_syncswapbuffers attribute and the vm_asyncswapbuffers attribute without rebooting the system. See Chapter 3 for information about modifying kernel subsystem attributes.

12.6 Reserving Physical Memory for Shared Memory

Granularity hints allow you to reserve a portion of physical memory at boot time for shared memory. This functionality allows the translation lookaside buffer to map more than a single page, and enables shared page table entry functionality, which may result in more cache hits.

On some database servers, using granularity hints provides a 2 to 4 percent run-time performance gain that reduces the shared memory detach time. See your database application documentation to determine if you should use granularity hints.

For most applications, use the Segmented Shared Memory (SSM) functionality (the default) instead of granularity hints.

To enable granularity hints, you must specify a value for the vm subsystem attribute gh_chunks. In addition, to make granularity hints more effective, modify applications to ensure that both the shared memory segment starting address and size are aligned on an 8-MB boundary.

Section 12.6.1 and Section 12.6.2 describe how to enable granularity hints.

12.6.1 Tuning the Kernel to Use Granularity Hints

To use granularity hints, you must specify the number of 4-MB chunks of physical memory to reserve for shared memory at boot time. This memory cannot be used for any other purpose and cannot be returned to the system or reclaimed.

To reserve memory for shared memory, specify a nonzero value for the gh_chunks attribute. For example, if you want to reserve 4 GB of memory, specify 1024 for the value of gh_chunks (1024 * 4 MB = 4 GB). If you specify a value of 512, you will reserve 2 GB of memory.

The value you specify for the gh_chunks attribute depends on your database application. Do not reserve an excessive amount of memory, because this decreases the memory available to processes and the UBC.

Note

If you enable granularity hints, disable the use of segmented shared memory by setting the value of the ipc subsystem attribute ssm_threshold attribute to 0.

You can determine if you have reserved the appropriate amount of memory. For example, you can initially specify 512 for the value of the gh_chunks attribute. Then, enter the following dbx commands while running the application that allocates shared memory:

```
# /usr/ucb/dbx -k /vmunix /dev/mem

(dbx) px &gh_free_counts
0xfffffc0000681748
(dbx) 0xfffffc0000681748/4X
fffffc0000681748:  0000000000000402 0000000000000004
fffffc0000681758:  0000000000000000 0000000000000002
(dbx)
```

The previous example shows:

- The first number (402) specifies the number of 512-page chunks (4 MB).
- The second number (4) specifies the number of 64-page chunks.
- The third number (0) specifies the number of 8-page chunks.
- The fourth number (2) specifies the number of 1-page chunks.

To save memory, you can reduce the value of the gh_chunks attribute until only one or two 512-page chunks are free while the application that uses shared memory is running.

The following vm subsystem attributes also affect granularity hints:

- The gh_min_seg_size — Specifies the shared memory segment size above which memory is allocated from the memory reserved by the gh_chunks attribute. The default is 8 MB.
- gh_fail_if_no_mem — When set to 1 (the default), the shmget function returns a failure if the requested segment size is larger than the value specified by the gh_min_seg_size attribute, and if there is insufficient memory in the gh_chunks area to satisfy the request.

  If the value of the gh_fail_if_no_mem attribute is 0, the entire request will be satisfied from the pageable memory area if the request is larger than the amount of memory reserved by the gh_chunks attribute.
- gh_keep_sorted — Specifies whether the memory reserved for granularity hints is sorted. The default does not sort reserved memory.
- gh_front_alloc — Specifies whether the memory reserved for granularity hints is allocated from low physical memory addresses (the default). This functionality is useful if you have an odd number of memory boards.

In addition, messages will display on the system console indicating unaligned size and attach address requests. The unaligned attach messages are limited to one per shared memory segment.

See Chapter 3 for information about modifying kernel subsystem attributes.

12.6.2 Modifying Applications to Use Granularity Hints

You can make granularity hints more effective by making both the shared memory segment starting address and size aligned on an 8-MB boundary.

To share third-level page table entries, the shared memory segment attach address (specified by the shmat function) and the shared memory segment size (specified by the shmget function) must be aligned on an 8-MB boundary. This means that the lowest 23 bits of both the address and the size must be 0.

The attach address and the shared memory segment size is specified by the application. In addition, System V shared memory semantics allow a maximum shared memory segment size of 2 GB minus 1 byte. Applications that need shared memory segments larger than 2 GB can construct these regions by using multiple segments. In this case, the total shared memory size specified by the user to the application must be 8-MB aligned. In addition, the value of the shm_max attribute, which specifies the maximum size of a System V shared memory segment, must be 8-MB aligned.

If the total shared memory size specified to the application is greater than 2 GB, you can specify a value of 2139095040 (or 0x7f800000) for the value of

the shm_max attribute. This is the maximum value (2 GB minus 8 MB) that you can specify for the shm_max attribute and still share page table entries.

Use the following dbx command sequence to determine if page table entries are being shared:

```
# /usr/ucb/dbx -k /vmunix /dev/mem

(dbx) p *(vm_granhint_stats *)&gh_stats_store
 struct {
     total_mappers = 21
     shared_mappers = 21
     unshared_mappers = 0
     total_unmappers = 21
     shared_unmappers = 21
     unshared_unmappers = 0
     unaligned_mappers = 0
     access_violations = 0
     unaligned_size_requests = 0
     unaligned_attachers = 0
     wired_bypass = 0
     wired_returns = 0
 }
 (dbx)
```

For the best performance, the shared_mappers kernel variable should be equal to the number of shared memory segments, and the unshared_mappers, unaligned_attachers, and unaligned_size_requests variables should be 0.

Because of how shared memory is divided into shared memory segments, there may be some unshared segments. This occurs when the starting address or the size is aligned on an 8-MB boundary. This condition may be unavoidable in some cases. In many cases, the value of total_unmappers will be greater than the value of total_mappers.

Shared memory locking changes a lock that was a single lock into a hashed array of locks. The size of the hashed array of locks can be modified by modifying the value of the vm subsystem attribute vm_page_lock_count. The default value is 0.

12.7 Improving Performance with Big Pages

Big pages memory allocation supports mapping a page of virtual memory to 8, 64, or 512 pages of physical memory. Given physical memory's current 8-KB page size, this means that a single page of virtual memory can map to 64, 512, or 4096 KB. Using big pages can minimize the performance penalties that are associated with misses in the translation lookaside buffer. The result can be improved performance for applications that need to map large amounts of data.

Unlike granularity hints, which reserve memory at boot time and can be used only with system V shared memory, big pages allocates memory at run

time and supports anonymous memory (for example, `mmap` and `malloc`) as well as System V shared memory, stack memory, and text segments.

Big pages memory allocation is most effective when used with memory-intensive applications, such as large databases, running on systems with robust physical memory resources. Systems with limited memory resources, and systems where the workload stresses memory resources, are not good candidates for using big pages. Similarly, if a system does not run memory-intensive applications that require large chunks of memory, it may not benefit from big pages.

12.7.1 Using Big Pages

Enabling and using big pages is controlled through the following attributes of the `vm` kernel subsystem:

`vm_bigpg_enabled` — Enable big pages

Enables (1) or disables (0) big pages.

Enabling big pages automatically disables granularity hints; `gh_chunks`, `rad_gh_regions`, and related attributes are ignored when `vm_bigpg_enabled` is set to 1.

When big pages is disabled, the associated `vm_bigpg*` attributes are ignored.

Default value: 0 (disabled)

Can be set only at boot time.

`vm_bigpg_thresh` — Apportion free memory among page sizes

The percentage of physical memory that should be maintained on the free page list for each of the four possible page sizes (8, 64, 512, and 4096 KB).

When a page of memory is freed, an attempt is made to coalesce the page with adjacent pages to form a bigger page. When an 8-KB page is freed, an attempt is made to coalesce it with 7 other such pages to form a 64-KB page. If that succeeds, the 64-KB page is now free and so an attempt is made to coalesce it with 7 other 64 KB pages to form a 512 KB page. This page is coalesced with 7 other 512 KB pages, if available, to form a 4 Mbyte page. The process stops there.

The `vm_bigpg_thresh` attribute sets the threshold at which coalescing of free memory for each page size begins. If `vm_bigpg_thresh` is 0 percent, then attempts to coalesce pages of size 8, 64, or 512 KB occur whenever a page of that size is freed. The result can be that all smaller pages are coalesced and free pages are all 4096 KB in size.

If vm_bigpg_thresh is 6 percent, the default, then attempts to coalesce pages of 8 KB occur only after 6 percent of system memory consists of 8 KB pages. The same holds for the larger page sizes. The result is 6 percent of free pages are 8 KB in size, 6 percent are 64 KB in size, 6 percent are 512 KB in size. The remaining free pages are 4096 KB in size. This assumes there is enough free memory to allocate 6 percent of system memory to 512-KB pages. When free memory gets low, allocation of free pages to the largest page size, 4096 KB, is affected first, then allocation to 512-KB pages, and last allocation to 64-KB page sizes.

With smaller values of vm_bigpg_thresh, more pages are coalesced, and so fewer pages are available at the smaller sizes. This can result in a performance degradation as a larger page will then have to be broken into smaller pieces to satisfy an allocation request for one of the smaller page sizes. If vm_bigpg_thresh is too large, fewer large size pages will be available and applications may not be able to take full advantage of big pages. Generally, the default value will suffice, but this value can be increased if the system work load requires more small pages.

Default value: 6 percent. Minimum value: 0 percent. Maximum value: 25 percent.

Can be set at boot time and run time.

12.7.2 Determining when a Memory Object uses Big Pages

The attributes that determine when a particular type of memory object uses big pages, vm_bigpg_anon, vm_bigpg_seg, vm_bigpg_shm, vm_bigpg_ssm, and vm_bigpg_stack, each has a default value of 64. This represents, in KB, the smallest amount of memory that a process can request and still benefit from an extended virtual page size.

For this default value of 64, the kernel handles a memory allocation request for 64 KB or greater by creating, depending on the size of the request, one or more virtual pages whose sizes can be a mix of 8 KB, 64 KB, 512 KB, and 4096 KB. The attribute value does not determine the page size. That is, the 64-KB default does not mean that all virtual pages are 64 KB in size. Instead, the kernel chooses a page size (or combination of sizes) that is most appropriate for the total amount of memory being requested and does so in the context of any alignment restrictions that the request might impose. The kernel handles memory allocation requests for fewer than 64 KB by using the default algorithm that maps one virtual page to 8 KB of physical memory.

Increasing the value of the attribute to greater than 64 restricts big pages memory allocation to a subset of the applications that might otherwise benefit from it. For example, setting an attribute to 8192 means that only

programs that request allocations of 8192 or more KB are allocated virtual pages larger than 8 KB.

Setting the value of vm_bigpg_anon, vm_bigpg_seg, vm_bigpg_shm, vm_bigpg_ssm, or vm_bigpg_stack to 0 disables big pages memory allocation for the type of memory object identified by the attribute. For example, setting vm_bigpg_anon to 0 disables big pages memory allocation for processes that request allocations of anonymous memory. There are no clear benefits to disabling big pages memory allocation for specific types of memory.

Changes to vm_bigpg_anon, vm_bigpg_seg, vm_bigpg_shm, vm_bigpg_ssm, or vm_bigpg_stack after the system is booted apply only to new memory allocations; run-time changes do not affect those memory mappings that are already in place.

Setting any of the following attributes to a value from 1 to 64 is the same as setting it to 64.

Note

Consult your support representative before changing any of the following per-object controls to values other than their default of 64 KB.

vm_bigpg_anon — Big pages for anonymous memory

Sets the minimum amount of anonymous memory (in KB) that a user process must request before the kernel maps a virtual page in the process address space to multiple physical pages. Anonymous memory is requested by calls to mmap(), nmmap(),malloc(), and amalloc(). Anonymous memory for memory mapped files is not supported.

Note

If the anon_rss_enforce attribute (which sets a limit on the resident set size of a process) is 1 or 2, it overrides and disables big pages memory allocation of anonymous and stack memory. Set anon_rss_enforce to 0 if you want big pages memory allocation for anonymous and stack memory.

Default value: 64 KB

Can be set at boot time and run time.

vm_bigpg_seg — Big pages for program text objects

Sets the minimum amount of memory (in KB) that a user process must request for a program text object before the kernel maps a virtual page in the process address space to multiple physical pages. Allocations for program text objects are generated when the process executes a program or loads a shared library. See also the descriptions of vm_segment_cache_max and vm_segmentation.

Default value: 64 KB

Can be set at boot time and run time.

vm_bigpg_shm — Big pages for shared memory

Sets the minimum amount of System V shared memory, in KB, that a user process must request before the kernel maps a virtual page in the process address space to multiple physical pages. Allocations for System V shared memory are generated by calls to shmget(), shmctl(), and nshmget().

Default value: 64 KB

Can be set at boot time and run time.

vm_bigpg_ssm — Big pages for segmented shared memory

Sets the minimum amount, in KB, of segmented shared memory (System V shared memory with shared page tables) that a user process must request before the kernel maps a virtual page in the process address space to multiple physical pages. Requests for segmented shared memory are generated by calls to shmget(), shmctl(), and nshmget().

The vm_bigpg_ssm attribute is disabled if the ssm_threshold IPC attribute is set to 0. The value of ssm_threshold must be equal to or greater than the value of SSM_SIZE. By default, ssm_threshold equals SSM_SIZE. See sys_attrs_ipc(5) for more information.

Default value: 64 KB

Can be set at boot time and run time.

vm_bigpg_stack — Big pages for stack memory

Sets the minimum amount of memory, in KB, needed for the user process stack before the kernel maps a virtual page in the process address space to multiple physical pages. Stack memory is automatically allocated by the kernel on the user's behalf.

If the anon_rss_enforce attribute (which sets a limit on the resident set size of a process) is 1 or 2, it overrides and disables big pages memory allocation of anonymous and stack memory. Set

anon_rss_enforce to 0 if you want big pages memory allocation for anonymous and stack memory.

Default value: 64 KB

Can be set at boot time and run time.

See sys_attrs_vm(5) for more information.

13

Managing CPU Performance

You can improve system performance by optimizing CPU resources. This chapter describes how to perform the following tasks:

- Obtain information about CPU performance by using the CPU monitoring tools (Section 13.1)
- Improve CPU performance by adding processors or running the class scheduler to control the execution of tasks and applications. (Section 13.2)

13.1 Monitoring CPU Performance Information

Table 13–1 describes the tools you can use to gather information about CPU usage.

Table 13–1: CPU Monitoring Tools

Name	Description	Reference
`ps`	Displays current statistics for running processes, including CPU usage, the processor and processor set, and the scheduling priority.	Section 12.3.2
`vmstat`	Displays information about process threads, virtual memory usage (page lists, page faults, page ins, and page outs), interrupts, and CPU usage (percentages of user, system, and idle times). First reported are the statistics since boot time; subsequent reports are the statistics since a specified interval of time.	Section 12.3.1

Table 13–1: CPU Monitoring Tools (cont.)

Name	Description	Reference
`uptime`	Displays the system load average and the number of jobs in the run queue for the last 5 seconds, the last 30 seconds, and the last 60 seconds. The `uptime` command also shows the number of users logged into the system and how long a system has been running.	Section 13.1.1
`(kdbx) cpustat`	Displays CPU statistics, including the percentages of time the CPU spends in various states.	Section 13.1.2
`(kdbx) lockstats`	Displays lock statistics for each lock class on each CPU in the system.	Section 13.1.3
`sys_check`	Analyzes system configuration and displays statistics and checks kernel variable settings and memory and CPU resources, and provides performance data and lock statistics for SMP and kernel profiles.	See Section 2.3.3 or see `sys_check`(8) for more information.
Process Tuner	Displays current statistics for running processes. Invoke the Process Tuner graphical user interface (GUI) from the CDE Application Manager to display a list of processes and their characteristics, display the processes running for yourself or all users, display and modify process priorities, or send a signal to a process. While monitoring processes, you can select parameters to view (percent of CPU usage, virtual memory size, state, and `nice` priority) and also sort the view.	Section 13.2.2.1.1

Table 13–1: CPU Monitoring Tools (cont.)

Name	Description	Reference
`monitor`	Collects a variety of performance data on a running system and either displays the information in a graphical format or saves it to a binary file.	See `monitor`(3) for more information.
`top`	Provides continuous reports on the state of the system, including a list of the processes using the most CPU resources.	The `top` command is available on the Tru64 UNIX Freeware CD-ROM. See ftp://eecs.nwu.edu/pub/top for information.
`ipcs -a`	Displays interprocess communication (IPC) statistics for currently active message queues, shared-memory segments, semaphores, remote queues, and local queue headers. The information provided in the following fields reported by the `ipcs` –a command can be especially useful: `QNUM`, `CBYTES`, `QBYTES`, `SEGSZ`, and `NSEMS`.	See `ipcs`(1) for more information.
`w`	Displays the current time, the amount of time since the system was last started, the users logged in to the system, and the number of jobs in the run queue for the last 5 seconds, 30 seconds, and 60 seconds.	See `w`(1) for more information.
`xload`	Displays the system load average in a histogram that is periodically updated.	See `xload`(1X) for more information.

The following sections describe the `ps`, `vmstat`, `uptime`, `cpustat`, and `lockstats` commands in detail.

13.1.1 Monitoring the Load Average by Using the uptime Command

The `uptime` command shows how long a system has been running and the load average. The load average counts the jobs that are waiting for disk I/O, and applications whose priorities have been changed with either the `nice` or the `renice` command. The load average numbers give the average number of jobs in the run queue for the last 5 seconds, the last 30 seconds, and the last 60 seconds.

An example of the uptime command is as follows:

```
# /usr/ucb/uptime
1:48pm  up 7 days,  1:07,  35 users,  load average: 7.12, 10.33, 10.31
```

The command output displays the current time, the amount of time since the system was last started, the number of users logged into the system, and the load averages for the last 5 seconds, the last 30 seconds, and the last 60 seconds.

From the command output, you can determine whether the load is increasing or decreasing. An acceptable load average depends on your type of system and how it is being used. In general, for a large system, a load of 10 is high, and a load of 3 is low. Workstations should have a load of 1 or 2.

If the load is high, look at what processes are running with the ps command. You may want to run some applications during offpeak hours. See Section 12.3.2 for information about the ps command.

You can also lower the priority of applications with the nice or renice command to conserve CPU cycles. See nice(1) and renice(8) for more information.

13.1.2 Checking CPU Usage by Using the kdbx Debugger cpustat Extension

The kdbx debugger cpustat extension displays CPU statistics, including the percentages of time the CPU spends in the following states:

- Running user-level code
- Running system-level code
- Running at a priority set with the nice function
- Idle
- Waiting (idle with input or output pending)

The cpustat extension to the kdbx debugger can help application developers determine how effectively they are achieving parallelism across the system.

By default, the kdbx cpustat extension displays statistics for all CPUs in the system. For example:

```
# /usr/bin/kdbx -k /vmunix /dev/mem
(kdbx)cpustat
 Cpu   User (%)    Nice (%) System (%)  Idle (%)   Wait (%)
===== ========== ========== ========== ========== ==========
    0       0.23       0.00       0.08      99.64       0.05
    1       0.21       0.00       0.06      99.68       0.05
```

See the *Kernel Debugging* manual and kdbx(8) for more information.

13.1.3 Checking Lock Usage by Using the kdbx Debugger lockstat Extension

The kdbx debugger lockstats extension displays lock statistics for each lock class on each CPU in the system, including the following information:

- Address of the structure
- Class of the lock for which lock statistics are being recorded
- CPU for which the lock statistics are being recorded
- Number of instances of the lock
- Number of times that processes have tried to get the lock
- Number of times that processes have tried to get the lock and missed
- Percentage of time that processes miss the lock
- Total time that processes have spent waiting for the lock
- Maximum amount of time that a single process has waited for the lock
- Minimum amount of time that a single process has waited for the lock

For example:

```
# /usr/bin/kdbx –k /vmunix /dev/mem
(kdbx)lockstats
```

See the *Kernel Debugging* manual and kdbx(8) for more information.

13.2 Improving CPU Performance

A system must be able to efficiently allocate the available CPU cycles among competing processes to meet the performance needs of users and applications. You can improve performance by optimizing CPU usage.

Table 13–2 describes the guidelines to improve CPU performance.

Table 13–2: Primary CPU Performance Improvement Guidelines

Guidelines	Performance Benefit	Tradeoff
Add processors (Section 13.2.1)	Increases CPU resources	Applicable only for multiproccessing systems, and may affect virtual memory performance
Use the class scheduler (Section 13.2.2)	Allocates CPU resources to critical applications	None
Prioritize job (Section 13.2.3)	Ensures that important applications have the highest priority	None

Table 13–2: Primary CPU Performance Improvement Guidelines (cont.)

Guidelines	Performance Benefit	Tradeoff
Schedule jobs at offpeak hours (Section 13.2.4)	Distributes the system load	None
Stop the `advfsd` daemon (Section 13.2.5)	Decreases demand for CPU power	Applicable only if you are not using the AdvFS graphical user interface
Use hardware RAID (Section 13.2.6)	Relieves the CPU of disk I/O overhead and provides disk I/O performance improvements	Increases costs

The following sections describe how to optimize your CPU resources. If optimizing CPU resources does not solve the performance problem, you may have to upgrade your CPU to a faster processor.

13.2.1 Adding Processors

Multiprocessing systems allow you to expand the computing power of a system by adding processors. Workloads that benefit most from multiprocessing have multiple processes or multiple threads of execution that can run concurrently, such as database management system (DBMS) servers, Internet servers, mail servers, and compute servers.

You may be able to improve the performance of a multiprocessing system that has only a small percentage of idle time by adding processors. See Section 12.3.1 for information about checking idle time.

Before you add processors, you must ensure that a performance problem is not caused by the virtual memory or I/O subsystems. For example, increasing the number of processors will not improve performance in a system that lacks sufficient memory resources.

In addition, increasing the number of processors may increase the demands on your I/O and memory subsystems and could cause bottlenecks.

If you add processors and your system is metadata-intensive (that is, it opens large numbers of small files and accesses them repeatedly), you can improve the performance of synchronous write operations by using a RAID controller with a write-back cache (see Section 9.4).

13.2.2 Using the Class Scheduler

The class scheduler provides you with a method of controlling the execution of tasks or applications by restricting the length of time that they can access the processor (CPU). For example, daemons such as the print spooler

are given less access time. The CPU will then have more time available to perform other tasks. To do this, you specify that the print daemon `/usr/lbin/lpd` is allowed to use no more than a certain percentage of the available CPU time. You can group resource user identifiers, such as a user's UID (user identification), into classes and assign the required CPU access time to each class.

This feature can help you to allocate system resources so that the most important work receives the required processing time. For example, you might want to run two versions of a production database on your system. One version is used as part of your business operations, while the other is a test copy, with different tuning parameters. You can assign the test database to a different class to prevent it from affecting your daily operations.

To set up and use the class scheduler, you must complete the following steps:

1. Plan the allocation of CPU resources.
2. Use `class_admin` to set up and maintain the class database.
3. Create classes and add members to the classes.
4. Verify class entries using the `show` command.
5. Save the entries to the database.
6. Enable class scheduling to start the daemon.

Use the class scheduler commands to monitor and control scheduling as follows:

- Execute `class_admin` commands such as `stat` from the command line or a shell script without running an interactive session
- Use the `runclass` command to execute a task according to the priorities set for a particular class

The following sections suggest a systematic approach to using class scheduling, although it is not necessary to perform tasks in a specific sequence. There are two methods of accessing the class scheduler:

- Manual

  By executing `class_admin` commands from the command line to configure a default database, add classes and class members, and enable the class scheduling daemon to create a quick fix to a CPU resource sharing problem.

- Graphical Interface

  By using the graphical user interface available as a SysMan Menu suboption, Class Scheduling, which is available under the Monitoring and Tuning menu option.

See the *System Administration* manual for information on running the SysMan Menu. Section 13.2.2.6 describes how you use the graphical interface. See the online help for additional information on valid data entries.

The following reference pages contain detailed information on using the class scheduler commands and options:

- `class_scheduling`(4)
- `class_admin`(8)
- `runclass`(1)
- `sysman`(8)

The following command displays online help for the `class_admin` command:

```
# /usr/sbin/class_admin help
```

13.2.2.1 Class Scheduler Overview

To use the class scheduler, you must first create a database file and populate the file with one or more classes. Each class is assigned a CPU value that controls its access to processing time, expressed as a percentage of the total CPU time availability. You can assign one or more applications or groups of applications to a class, identified according to a unique system process identifier such as:

- UID — User identifier, a unique number assigned to each user account (login)
- GID — Group identifier, a number or name assigned to several user accounts to indicate that they belong to the same group
- PID — Process identifier, a system-assigned number that is unique to each process
- PGID — Process group identifier, a system-assigned number that is unique to each process group
- SESS — Session identifier, a system-assigned number that is unique to each session

The PID, PGID, and SESS identifiers are usually temporary and do not persist across a reboot, ceasing to exist when a task is completed. They are not stored in the database and have no effect when the system or task is restarted.

After the database is established, you can enable class scheduling. This operation starts a class scheduling daemon and puts the CPU access restrictions into effect. Other commands enable you to review classes, change

contents or scheduling parameters, and delete components or entire classes. When a class scheduling database is configured and enabled, you can:

- Use `runclass` to execute a task (process) according to the CPU access value set for a specific class. For example, you might set a value for interactive operations that is much higher than background processes such as print daemons. To temporarily use the higher value for a print job, you can execute the `lpr` command in the same class as interactive operations.
- Use the `class_admin` command to execute class scheduling commands from within scripts.

13.2.2.1.1 Related Utilities

The following utilities are also available for use when monitoring and tuning processes:

- The `nice` command
- The Process Tuner (`dxproctuner`) graphical interface, available from the CDE MonitoringTuning folder in the Application Manager - System_Admin
- You can invoke the `iostat` and `vmstat` commands from the SysMan Menu

13.2.2.1.2 Invoking the Class Scheduler

The class scheduler is provided as both a command-line interface and a graphical user interface. You can invoke the class scheduler in several ways, depending on which user environment you are working from:

- From the SysMan Menu, select the Monitoring and Tuning branch, then select the Class Scheduling task.
- From the command line, enter either of the following commands:

```
# sysman class_sched

# sysman -menu "Class Scheduling"
```

- From CDE (assuming your system is running a graphics environment with CDE), follow these steps:

  1. Select the Application Manager from the CDE front panel.
  2. Select the System_Admin Software Management Group.
  3. Select the Configuration Software Management Group.
  4. Select the class scheduler icon.

The following sections focus on using the command-line method, and provide a brief introduction to using the graphical interface. See the online help for more information in using the graphical interface.

13.2.2.2 Planning Class Scheduling

How you allocate CPU resources depends on your system environment and what resources and priorities must be considered. A typical scenario is to assign a higher CPU percentage to interactive tasks so that users do not encounter long response times. Most batch or background processes will be assigned a lower CPU percentage, while some specific background processes might require a higher CPU percentage. For example, if a nightly backup is being performed, you might not want it to have such a low CPU percentage that it does not complete in a reasonable time.

If your system is involved with critical real-time tasks that must take precedence over interactive processes, your course of action might be different. In such cases you should design a baseline that assigns processes to classes. You can then monitor processes and gather user feedback to tune the database by moving tasks from class to class or by changing the CPU access time of the classes.

Do not use the root account to create test processes when you configure class scheduling. Root account processes always take precedence over others, even when assigned to an existing restricted class.

13.2.2.3 Configuring Class Scheduling

Use the `class_admin` command to configure an initial database. This command provides:

- An interactive command (with subcommands) that enables you to create and administer a database of classes. The database is stored in the binary file `/etc/class`, which cannot be manually edited. Type `help` at the `class>` command prompt for a list of options.
- A command mode that allows you to execute `class_admin` commands at the command prompt, or include commands in shell scripts.

A database must be configured before you can enable class scheduling with the `enable` command. If a database does not exist when you enter the `class_admin` command, the command will invoke an interactive session and prompt you to automatically configure a database. If a script invokes the `class_admin` command, it uses the system defaults to configure the database.

The following example shows an interactive configuration session using `class_admin`. In the actual output, the lines will be formatted to fit in 80 columns.

```
# /usr/sbin/class_admin
                    Class Scheduler Administration

configure:

Shall processes that have not been explicitly
assigned to a defined class be assigned to a
'default' class?  Enter (yes/no) [no]: yes

Enforce class scheduling when the CPU is otherwise
idle? (yes/no) [no]: yes

How often do you want the system to reset class usage?
Enter number of seconds (1): 2
class>
```

The configuration values have the following effect:

- To be scheduled, a process must be assigned to a class. If you answer `yes` to the first prompt, a special class called the `default` class is created. Any process that is not explicitly assigned to a defined class will be assigned to the default class.

  If you answer `no` to this prompt, then only those processes that are explicitly assigned to a defined class will be class scheduled.

- If you answer `yes` to the second prompt, you allow classes to exceed their allotted CPU time percentage when the system is otherwise idle. If you answer `no`, classes are restricted to their allotted percentage even if the CPU has no other work.

- The third prompt allows you to set the standard reset time for all classes. For example, if you choose the short default time of 1 second, each class will have more frequent, but shorter, opportunities to access the CPU.

  Use a small number (several seconds) if there are interactive jobs subject to class scheduling to give them a quick response time. If only batch jobs are class scheduled, response time is not an issue and you can specify larger values.

In the example, a default class was created and all current processes were assigned to that class. Class scheduling will be enforced even when the CPU is idle and class usage will be reset every five seconds.

To review the current configuration, enter the `show` command:

```
class> show
Configuration:
 -Processes not explicitly defined in the database are
  class scheduled.
 -If the processor has some idle time, class scheduled
  processes are not allowed to exceed their cpu percentage.
 -The class scheduler will check class CPU usage every 2
  seconds.
```

```
Class scheduler status: disabled  current database: /etc/class

Classes:

 default targeted at 100%:
    class members:
    Every one not listed below
```

The next step in the process is to create classes and populate the classes with system processes such as tasks, daemons, or user accounts by using the appropriate identifiers such as UID or SESS.

13.2.2.4 Creating and Managing Classes

When the database is configured, you can administer classes as follows:

- Create a class:
  - Add processes to the class
  - Delete processes from a class
- Change the CPU access value (time percentage) of any class
- Destroy an entire class, whether empty or populated
- Show details of class members and configuration settings
- View statistics of actual CPU use against current priority settings

Some of these options are described briefly in the following sections. For detailed descriptions of command options, see the online help and reference pages.

13.2.2.4.1 Creating a Class

To create a class, either use the command-mode or enter an interactive session as follows:

```
# class_admin
class> create high_users 50
```

The command-mode version is entered as follows:

```
# class_admin create batch_jobs 10
batch_jobs created at 10% cpu usage


changes saved
```

The first command creates a class named `high_users` and assigns a CPU usage restriction of 50 percent. The second command creates a class named `batch_jobs` and assigns a CPU usage restriction of 10 percent. In command mode, the changes are automatically saved to the database in `/etc/class`. When making changes to classes interactively, use the `save` command to commit changes to the database. If you attempt to end the session with the

quit command and there are unsaved changes, you will be prompted to save or discard the changes before quitting the interactive session as follows:

```
class> quit
Class scheduler database modified.
Save changes? (yes/no) [yes]:yes

changes saved
```

13.2.2.4.2 Managing Identifier Types Within Classes

Unique system-assigned identifiers that the class scheduler recognizes (such as the PID, GID, or UID) identify which processes are members of a specific class. After you have created classes, you can add UIDs and GIDs or processes to one or more classes by using the add command. You must specify the type of identifier (ID) used and enter one or more unique identifiers. UIDs and GIDs can be determined from the /etc/passwd and /etc/group files. Alternatively, you can use the graphical interface Account Manager (dxaccounts) to display UID and Group information.

Process identifiers can be obtained from system files or by using a command such as ps. With the ps command, you can determine the values of PID, PGID, and SESS. Enter the following command to display the PID for every process running on the system:

```
# /sbin/ps aj


USER   PID PPID  PGID  SESS JOBC S    TTY          TIME COMMAND
walt  5176 5162  5176  2908   1  S    ttyp1    0:01.30 -sh (csh)
root 12603 5176 12603  2908   1  R  + ttyp1    0:00.05 ps aj
```

See ps(1) for more information and a definitive list of the process data items displayed when you use this command.

The following identifiers are supported:

gid

A group identification number from the /etc/group file. For example, if you are adding members to a class, using this number will add all the users that are assigned to the group.

uid

A user identification number from the /etc/passwd file. For example, if you are adding members to a class, this number will add only the specific user to which the UID is assigned.

pgrp

A process group identifier. In the output from the ps aj command, see the entries under the PGID table heading in the previous example.

session

A session identifier. In the output from the ps aj command, see the entries under the SESS table heading in the previous example.

pid

The process identifier. In the output from the ps aj command, see the entries under the PID table heading in the previous example.

It is most likely that you will use types uid and gid in your established classes, as these values will persist across a reboot or when class scheduling is stopped and restarted. You can use the account management tools, such as dxaccounts or the Accounts option of the SysMan Menu, to list UIDs and GIDs for users and groups. The identifiers associated with types pgrp, session, and pid are temporary, and will not exist on reboot or when a process terminates.

13.2.2.4.3 Enabling the Class Scheduler

To enable the class scheduler daemon, enter the following command:

```
# class_admin enable
Class scheduling enabled and daemon \
/usr/sbin/class_daemon started.
```

To disable the daemon, enter the following command:

```
# class_admin disable
Class scheduling disabled.
```

13.2.2.4.4 Adding Members to a Class

To add a process to a class, enter the add command as shown in the following interactive mode example:

```
class> add batch_jobs uid 234 457 235
```

You must use one of the unique identifiers previously specified, and you cannot add the same identifier to a class more than once. The same procedure can be performed in command mode or from a script as follows:

```
# class_admin add batch_jobs uid 234 457 235
uid 234 457 235 added to high_users
```

In command mode, additions to a class are automatically saved to the /etc/class database.

13.2.2.4.5 Deleting Members From a Class

To delete one or more processes from a class, enter the `delete` command in interactive or command mode. For example:

```
class> delete high_users uid 11
uid 11 deleted from high_users
```

This example deletes the single UID number 11 from class `high_users`.

13.2.2.4.6 Other Class Management Options

See `class_admin`(8) for information on the following options:

- Change the priority of a class. For example:

```
class> change batch_jobs 20
batch_jobs retargeted at 20%
```

- Destroy an entire class, whether empty or full. For example:

```
class> destroy high_users
high_users is not empty.
        to destroy anyway? [yes/no]:yes
        high_users destroyed
```

- Loading and saving scheduling databases. For example:

```
class> load database_performance
current database modified and not saved
load new database anyway (destroys changes)? (yes/no) [yes]: \
yes
database database_performance loaded
```

  In this example the presence of unsaved modifications to the current database was detected, and the user was prompted to save the changes.

- View statistics of actual CPU use against current priority settings. For example:

```
class> stats
   Class scheduler status: enabled

class name   target percentage  actual percentage
  high_users      50%               40.0%
  batch_jobs      10%                2.0%
```

13.2.2.5 Using the runclass Command

Once you have established scheduler classes and enabled class scheduling, you can use the `runclass` command to execute a command in a particular class. When you use the `runclass` command as root user (superuser), your processes have unrestricted access to CPU resources even when assigned to an existing class. By default, root processes are never restricted. This ensures that no user process can lock up resources needed by the root account. If you need to test a class scheduler configuration, ensure that you

log in and create processes using a nonprivileged user account. You might want to set up dummy user accounts to perform such testing.

The following command uses the `runclass` command to open a terminal window and assign it to the previously created `high_users` class:

```
# runclass high_users xterm
```

The following command shows that the `pgrp` number for the terminal process is now identified as a member of that class:

```
# class_admin show
.
.
.
class members:
pgrp 24330     pgrp 24351     pgrp 24373
```

In this example, the identifier for the `xterm` process is added to the class. You can use the following command to view the running process:

```
# ps agx | grep xterm
```

See `runclass`(1) for more information.

13.2.2.6 Using the Class Scheduling Graphical Interface

The class scheduler can be launched from the SysMan menu by selecting the `Class Scheduling` option from the `Monitoring and Tuning` tasks. Alternatively, you can launch it from the Common Desktop Environment (CDE) Application Manager.

As for the command-line method of using the class scheduler described in preceding sections, the steps involved in initial configuration are as follows:

1. Plan your classes and the processes, users, or groups that will be in each class.
2. Configure and name a database by creating classes and adding them to the database.
3. Define the new database as the current database.
4. Start the class scheduling daemon.

You can complete these steps by using the SysMan Menu Class Scheduling main menu option, where the following three suboptions are available:

Configure Class Scheduler

This is the main option that you use to configure and initialize class scheduling. When you select this option, a window is displayed: Configure Class Scheduler on *hostname*. From here you can select one of the following options:

- Make Current... — Use this option to choose an existing database and make it the current database. When the system is first used,

only the default database is available from the option list. This database is a placeholder and contains no classes. You can modify the default or create new databases, adding options to the list.

- New... — Use this option to create a new database and add it to the list of optional databases. A data entry window will be displayed for you to name the database and select or create classes.
- Copy... — Use this option to copy an existing database to a file so that you can use it as a starting point for a new database. You will be prompted to enter a file name and location for the copy.
- Modify... — Use this option to change the configuration of an existing database. If you want to preserve the original database before modifying it, use the Copy... option first.
- Delete — Use this option to remove databases from the option list. You will not be able to recover these databases once removed.

The New... option is the main option and the one most frequently used. It is described in detail in Section 13.2.2.7. The Modify... option provides an identical interface, which allows you to change existing classes and databases.

The remaining menu options require only a confirmation and do not involve extensive data entry. For example, if you want to delete a database, you will only be prompted to confirm that the database is to be destroyed.

[Re]Start Class Scheduler

Use this option to start the class scheduling daemon, or restart it if it was stopped. You will be prompted to confirm your selection.

Stop Class Scheduler

Use this option to stop the class scheduling daemon. You will be prompted to confirm your selection.

See the *System Administration* manual for information on running the SysMan Menu. Section 13.2.2.6 describes how you use the graphical interface. See the online help for additional information on valid data entries.

13.2.2.7 Creating or Modifying a Database

When you select the New... or Modify... options, a screen is displayed titled Configure Class Scheduler: Create/Modify Scheduling Database. To create a new database, follow these steps:

1. In the Name: field, type the name of the database that you want to create. The name should reflect the function of the database, so that you can easily recognize it when it is displayed in a list of many options. For example, `served_applications`.
2. From the option list titled Available Scheduled Classes, you can select any existing classes. If you are setting up the first database, no classes will be listed and only the New.. option will be available for selection.
3. To create a new class, press the New... button to display the window titled Create a new class. In this window, you complete the following steps:
   - Enter a name for the class in the Class name field. The name should enable you to easily recognize the members of the class. For example, `principal_users`.
   - Move the slider bar adjacent to the CPU allocation label to assign a value for the percentage of CPU time allocated to this class.
   - From the pull-down menu in the Member type field, select the type of identifier you will use to allocate processes to this class. Only the Group ID and User ID will persist across reboots. Session, Process group, and Process ID identifiers will not persist.
   - In the member field, enter the name of the user from the `/etc/passwd` file, a group from the `/etc/group` file, or a process identifier from the output of the following command:

     ```
     # /sbin/ps aj
     ```

   - Select the OK button to complete the class entry and return to the previous window, or select the Apply button to complete this entry and retain the window to create further classes. Use the Cancel button if you do not want to proceed with the creation of a class.
4. When classes are created, they appear as entries in the optional list of Available Scheduling Classes. Apart from the class name, the CPU time percentage allocation and member and type are also displayed. You can now select classes to add to the database as follows:
   - Click on a class to highlight it
   - Press the Select button to add the class to the database.

5. When all required classes are selected, press the OK button to create the new database. The new database will be added to the list of Available Scheduling Databases.

You also use the Configure Class Scheduler: Create/Modify Scheduling Database window to perform maintenance and administrative operations on classes as follows:

- Use the Copy... option to copy a class and use it as the base for a new class.
- Use the Modify... option to change characteristics of a class.
- Use the Delete... option to destroy a class and remove it permanently from the Available Scheduling Classes.

To begin using the newly created database, follow these steps:

1. If the window titled Configure Class Scheduler on *hostname* is not already displayed, invoke the SysMan Menu and select the Configure Class Scheduler option.
2. Highlight the required database by clicking on it, then press the Make Current... button. You will be prompted to confirm or cancel your choice.
3. Press the OK button to return to the SysMan Menu, Class Scheduling options, and select the option titled [Re]Start Class Scheduler. You will be prompted to confirm your choice.

On completing these steps, the class scheduling daemon starts and uses the scheduling database that you specified. To verify and monitor that the database is working as anticipated, use the `show command` at the terminal command line. For example, to view scheduling statistics, enter the following command:

```
# class_admin stats

Class scheduler status: enabled \
current database: /etc/.cl_lab1

class name     target percentage  actual percentage
prio-tasks-lab        10                 10
```

You might need to spend some time monitoring tasks and system performance, tuning your classes to obtain the required results.

13.2.3 Prioritizing Jobs

You can prioritize jobs so that important applications are run first. Use the `nice` command to specify the priority for a command. Use the `renice` command to change the priority of a running process.

See `nice`(1) and `renice`(8) for more information.

13.2.4 Scheduling Jobs at Offpeak Hours

You can schedule jobs so that they run at offpeak hours (use the `at` and `cron` commands) or when the load level permits (use the `batch` command). This can relieve the load on the CPU and the memory and disk I/O subsystems.

See `at`(1) and `cron`(8) for more information.

13.2.5 Stopping the advfsd Daemon

The `advfsd` daemon allows Simple Network Management Protocol (SNMP) clients such as Netview to request AdvFS file system information. If you are not using the AdvFS graphical user interface (GUI), you can free CPU resources and prevent the `advfsd` daemon from periodically scanning disks by stopping the `advfsd` daemon.

To prevent the `advfsd` daemon from starting at boot time, rename `/sbin/rc3.d/S53advfsd` to `/sbin/rc3.d/T53advfsd`.

To immediately stop the daemon, enter the following command:

```
# /sbin/init.d/advfsd stop
```

13.2.6 Using Hardware RAID to Relieve the CPU of I/O Overhead

RAID controllers can relieve the CPU of the disk I/O overhead, in addition to providing many disk I/O performance-enhancing features. See Section 9.4 for more information about hardware RAID.

Glossary

This glossary lists the terms that are used to describe Tru64 UNIX performance, availability, and tuning.

active list
Pages that are being used by the virtual memory subsystem or the UBC.

adaptive RAID 3/5
See *dynamic parity RAID*

AL_PA
The Arbitrated Loop Physical Address (AL_PA) is used to address nodes on the Fibre Channel loop. When a node is ready to transmit data, it transmits Fibre Channel primitive signals that include its own identifying AL_PA.

anonymous memory
Modifiable memory that is used for stack, heap, or `malloc`.

arbitrated loop
A Fibre Channel topology in which frames are routed around a loop set up by the links between the nodes in the loop. All nodes in a loop share the bandwidth, and bandwidth degrades slightly as nodes and cables are added.

attributes
Dynamically configurable kernel variables, whose values you can modify to improve system performance. You can utilize new attribute values without rebuilding the kernel.

bandwidth
The rate at which an I/O subsystem or component can transfer bytes of data. Bandwidth is especially important for applications that perform large sequential transfers.

See also *transfer rate*

bitfile metadata table (BMT)
The bitfile metadata table describes the file extents on the volume.

blocking queue
The blocking queue is a queue in which reads and synchronous write requests are cached. The blocking queue is used primarily for reads and for kernel synchronous write requests.

See also *flush queue*

BMT
See *bitfile metadata table (BMT)*

bottleneck
A system resource that is being pushed near to its capacity and is causing a performance degradation.

bus extenders
Bus extenders are used by the UltraSCSI technology to configure systems and storage over long distances.

See also *bus segments*

bus segments
Bus segments are used by the UltraSCSI technology to configure systems and storage over long distances.

See also *bus extenders*

cache
A temporary location for holding data that is used to improve performance by reducing latency. CPU caches and secondary caches hold physical addresses. Disk track caches and write-back caches hold disk data. Caches can be volatile (that is, not backed by disk data or a battery) or nonvolatile.

cache hit
Data found in a cache.

cache hit rate
The measure of effective cached data.

cache miss
Data that was not found in a cache.

capacity
The maximum theoretical throughput of a system resource, or the maximum amount of data, in bytes, that a disk can contain. A resource that has reached its capacity may become a bottleneck and degrade performance.

cascaded switches
Multiple switches that may be connected to each other to form a network of switches.

See also *meshed fabric*

cluster
A loosely coupled group of servers (cluster member systems) that share data for the purposes of high availability. Some cluster products utilize a high-performance interconnect for fast and dependable communication.

Compaq Analyze
A diagnostic tool that provides error event analysis and translation.

copy-on-write page fault
A page fault that occurs when a process needs to modify a read-only virtual page.

configuration
The assemblage of hardware and software that comprises a system or a cluster. For example, CPUs, memory boards, the operating system, and mirrored disks are parts of a configuration.

configure
To set up or modify a hardware or software configuration. For example, configuring the I/O subsystem can include connecting SCSI buses and setting up mirrored disks.

data path
Determines the actual bandwidth for a bus.

deferred mode
A swap space allocation mode by which swap space is not reserved until the system needs to write a modified virtual page to swap space. Deferred mode is sometimes referred to as lazy mode.

delay
See *latency*

disk access time
A combination of the seek time and the rotational latency, measured in milliseconds. A low access time is especially important for applications that perform many small I/O operations.

See also *rotational latency*, *seek time*

disk partitions
Disk partitions are logical divisions of a disk that allow you to organize files by putting them into separate areas of varying sizes. Partitions hold data in structures called file systems and can also be used for system operations such as paging and swapping.

disk quotas
Allows the system administrator to limit the disk space available to users and to monitor disk space usage.

dynamically wired memory
Wired memory that is used for dynamically allocated data structures, such as system hash tables. User processes also allocate dynamically wired

memory for address space by using virtual memory locking interfaces, including the `mlock` function.

dynamic parity RAID

Also called adaptive RAID3/5, dynamic parity RAID combines the features of RAID3 and RAID5 to improve disk I/O performance and availability for a wide variety of applications. Adaptive RAID3/5 dynamically adjusts, according to workload needs, between data transfer-intensive algorithms and I/O operation-intensive algorithms.

eager mode

See *immediate mode*

extent

Contiguous area of disk space that AdvFS allocates to a file.

E_Port

Communication between two switches which is routed between two expansion ports.

fabric

A switch, or multiple interconnected switches, that route frames between the originator node (transmitter) and destination node (receiver).

fail over / failover

To automatically utilize a redundant resource after a hardware or software failure, so that the resource remains available. For example, if a cluster member system fails, the applications running on that system automatically fail over to another member system.

Fast SCSI

Enables I/O devices to attain high peak-rate transfers in synchronous mode.

Fast10

See *Fast SCSI*

Fast20

See *UltraSCSI*

FC-AL

See *arbitrated loop*

file-backed memory

Memory that is used for program text or shared libraries.

flush queue

The flush queue is a queue in which reads and synchronous write requests are cached. The flush queue is used primarily for buffer write requests or synchronous writes.

See also *blocking queue*

frame

All data is transferred in a packet of information called a frame. A frame is limited to 2112 bytes. If the information consists of more than 2112 bytes, it is divided up into multiple frames.

free list

Pages that are clean and are not being used (the size of the free list controls when page reclamation occurs).

F_Port

The ports within the fabric (fabric port). This port is called an F_port. Each F_port is assigned a 64-bit unique node name and a 64-bit unique port name when it is manufactured. Together, the node name and port name make up the worldwide name.

FL_Port

An F_Port containing the loop functionality is called an FL_Port.

hard zoning

Zones are enforced at the physical level across all fabric switches by hardware blocking of the Fibre Channel frames.

hardware RAID

A storage subsystem that provides RAID functionality by using intelligent controllers, caches, and software.

high availability

The ability of a resource to withstand a hardware or software failure. High availability is achieved by using some form of resource duplication that removes single points of failure. Availability also is measured by a resource's reliability. No resource can be protected against an infinite number of failures.

immediate mode

A swap space allocation mode by which swap space is reserved when modifiable virtual address space is created. Immediate mode is often referred to as eager mode and is the default swap space allocation mode.

inactive pages

The oldest pages that are being used by processes.

interprocess communication

The interprocess communication (IPC) is the exchange of information between two or more processes.

IPC

See *interprocess communication*

kernel variables
Variables that determine kernel and subsystem behavior and performance. System attributes and parameters are used to access kernel variables.

latency
The amount of time to complete a specific operation. Latency is also called delay. High performance requires a low latency time. I/O latency can be measured in milliseconds, while memory latency is measured in microseconds. Memory latency depends on the memory bank configuration and the system's memory requirements.

lazy mode
See *deferred mode*

lazy queue
Logical series of queues in which asynchronous write requests are cached.

link
The physical connection between an N_Port and another N_Port or an N_Port and an F_Port. A link consists of two connections, one to transmit information and one to receive information. The transmit connection on one node is the receive connection on the node at the other end of the link. A link may be optical fiber, coaxial cable, or shielded twisted pair.

mesh
See *meshed fabric*

meshed fabric
A cascaded switch configuration, which allows for network failures up to and including the switch without losing a data path to a SAN connected node.

mirroring
Maintaining identical copies of data on different disks, which provides high data availability and improves disk read performance. Mirroring is also known as RAID 1.

multiprocessor
A system with two or more processors (CPUs) that share common physical memory.

namei cache
Location where the virtual file system (VFS) caches a recently accessed file name and its corresponding vnode.

NetRAIN
A Redundant Array of Independent Netowork Adaptors interface provides a mechanism to protect against certain kinds of network connectivity failures.

network adapter
See *network interface card (NIC)*

network interface
See *network interface card (NIC)*

network interface card (NIC)
A circuit board used to create a physical connection to a network. A NIC is also called a network adapter or a network interface.

node
The source and destination of a frame. A node may be a computer system, a redundant array of independent disks (RAID) array controller, or a disk device. Each node has a 64-bit unique node name (worldwide name) that is built into the node when it is manufactured.

N_Port
Each node must have at least one Fibre Channel port from which to send or receive data. This node port is called an N_Port. Each port is assigned a 64-bit unique port name (worldwide name) when it is manufactured. An N_Port is connected directly to another N_Port in a point-to-point topology. An N_Port is connected to an F_Port in a fabric topology.

NL_Port
In an arbitrated loop topology, information is routed around a loop. A node port that can operate on the loop is called an NL_Port (node loop port). The information is repeated by each NL_Port until it reaches its destination. Each port has a 64-bit unique port name (worldwide name) that is built into the node when it is manufactured.

page
The smallest portion of physical memory that the system can allocate (8 KB of memory).

pageable memory
Physical memory that is not wired.

page coloring
The attempt to map a process' entire resident set into the secondary cache.

page fault
An instruction to the virtual memory subsystem to locate a requested page and make the virtual-to-physical address translation in the page table.

page in
To move a page from a disk location to physical memory.

page-in page fault
A page fault that occurs when a requested address is found in swap space.

page out
To write the contents of a modified (dirty) page from physical memory to swap space.

page table
An array containing an entry for each current virtual-to-physical address translation.

paging
The process by which pages that are allocated to processes and the UBC are reclaimed for reuse.

See also *Unified Buffer Cache*

parallel SCSI
The most common type of SCSI which supports SCSI variants that provide a variety of performance and configuration options.

parameters
Statically configurable kernel variables, whose values can be modified to improve system performance. You must rebuild the kernel to utilize new parameter values. Many parameters have corresponding attributes.

parity RAID
A type of RAID functionality that provides high data availability by storing on a separate disk or multiple disks redundant information that is used to regenerate data. Parity RAID is also knows as a type of RAID3.

physical memory
The total capacity of the memory boards installed in your system. Physical memory is either wired or it is shared by processes and the UBC.

preferred transfer size
Value of data transfer to and from the disk in sizes that are most efficient for the device driver. This value is provided by the device driver.

Privileged Architecture Library (PAL)
Controls the movement of addresses and data among the CPU cache, the secondary and tertiary caches, and physical memory. This movement is transparent to the operating system.

RAID
RAID (redundant array of independent disks) technology provides high disk I/O performance and data availability. The Tru64 UNIX operating system provides RAID functionality by using disks and the Logical Storage Manager software (LSM). Hardware-based RAID functionality is provided by intelligent controllers, caches, disks, and software.

RAID0

Also known as disk striping, RAID0 functionality divides data into blocks and distributes the blocks across multiple disks in a array. Distributing the disk I/O load across disks and controllers improves disk I/O performance. However, striping decreases availability because one disk failure makes the entire disk array unavailable.

RAID1

Also known as data mirroring, RAID1 functionality maintains identical copies of data on different disks in an array. Duplicating data provides high data availability. In addition, RAID1 improves the disk read performance, because data can be read from two locations. However, RAID1 decreases disk write performance, because data must be written twice. Mirroring *n* disks requires 2*n* disks.

RAID3

RAID3 functionality divides data blocks and distributes (stripes) the data across a disk array, providing parallel access to data. RAID3 provides data availability; a separate disk stores redundant parity information that is used to regenerate data if a disk fails. It requires an extra disk for the parity information. RAID3 increases bandwidth, but it provides no improvement in the throughput. RAID3 can improve the I/O performance for applications that transfer large amounts of sequential data.

RAID5

RAID5 functionality distributes data blocks across disks in an array. Redundant parity information is distributed across the disks, so each array member contains the information that is used to regenerate data if a disk fails. RAID5 allows independent access to data and can handle simultaneous I/O operations. RAID5 provides data availability and improves performance for large file I/O operations, multiple small data transfers, and I/O read operations. It is not suited to applications that are write-intensive.

random access pattern

Refers to an access pattern in which data is read from or written to blocks in various locations on a disk.

raw I/O

I/O to a disk or disk partition that does not use a file system. Raw I/O bypasses buffers and caches, and can provide better performance than file system I/O.

redundancy

The duplication of a resource for purposes of high availability. For example, you can obtain data redundancy by mirroring data across different disks or by using parity RAID. You can obtain system redundancy by setting up a cluster, and network redundancy by using multiple network connections.

The more levels of resource redundancy you have, the greater the resource availability. For example, a cluster with four member systems has more levels of redundancy and thus higher availability than a two-system cluster.

reliability

The average amount of time that a component will perform before a failure that causes a loss of data. Often expressed as the mean time to data loss (MTDL), the mean time to first failure (MTTF) or the mean time between failures (MTBF).

resident set

The complete set of all the virtual addresses that have been mapped to physical addresses (that is, all the pages that have been accessed during process execution).

resource

A hardware or software component (such as the CPU, memory, network, or disk data) that is available to users or applications.

rotational latency

The amount of time, in milliseconds, for a disk to rotate to a specific disk sector.

route

The path a packet takes through a network from one system to another. It enables you to commmunicate with other systems on other networks. Routes are stored on each system in the routing tables or routing database.

scalability

The ability of a system to utilize additional resources with a predictable increase in performance, or the ability of a system to absorb an increase in workload without a significant performance degradation.

scalable

A system's ability to utilize additional hardware resources with a predictable impact on performance.

SCSI

Small Computer System Interface (SCSI) is a device and interconnect technology.

SCSI bus speed

See *bandwidth*

seek time

The amount of time, in milliseconds, for a disk head to move to a specific disk track.

selective storage presentation (SSP)
Controls which server will have access to each storage unit. SSP also controls access at the storage unit level.

sequential access pattern
Refers to an access pattern in which data is read from or written to contiguous blocks on a disk.

serial SCSI
Reduces parallel SCSI's limitation on speed, distance, and connectivity, and also provides availability features like hot swap and fault tolerance. Serial SCSI is the next generation of SCSI.

short page fault
A page fault that occurs when a requested address is found in the virtual memory subsystem's internal data structures.

simple name server (SNS)
A switch service that stores names, addresses, and attributes for up to 15 minutes, and provides them to other devices in the fabric. SNS is defined by fibre channel standards and exists at a well known address. May also be referred to as a directory service.

SMP
Symmetrical multiprocessing (SMP) is the ability of a multiprocessor system to execute the same version of the operating system, access common memory, and execute instructions simultaneously.

soft zoning
A software implementation that is based on the Simple Name Server (SNS), enforcing a zone. It also works if all hosts honor it; it does not work if a host is not programmed to allow for soft zoning.

software RAID
Storage subsystem that provides RAID functionality by using software (for example, LSM).

static wired memory
Wired memory that is allocated at boot time and used for operating system data and text and for system tables, static wired memory is also used by the metadata buffer cache, which holds recently accessed UNIX file system (UFS) and CD-ROM file system (CDFS) metadata.

striping
Distributing data across multiple disks in a disk array, which improves I/O performance by allowing parallel access. Striping is also known as RAID 0. Striping can improve the performance of sequential data transfers and I/O operations that require high bandwidth.

swap device
A block device in a configured section of a disk.

swap in
To move a swapped-out process' pages from disk swap space to physical memory in order for the process to execute. Swapins occur only if the number of pages on the free page list is higher than a specific amount for a period of time.

swap out
To move all the modified pages associated with a low-priority process or a process with a large resident set size from physical memory to swap space. A swapout occurs when number of pages on the free page list falls below a specific amount for a period of time. Swapouts will continue until the number of pages on the free page list reaches a specific amount.

swap-space interleaving
See *striping*

swapping
Writing a suspended process' modified (dirty) pages to swap space, and putting the clean pages on the free list. Swapping occurs when the number of pages on the free list falls below a specific threshold.

switch zoning
Controls which server can communicate with each other and each storage controller host port. Switch zoning also controls access at the storage system level.

throughput
The rate at which an I/O subsystem or component can perform I/O operations. Throughput is especially important for applications that perform many small I/O operations.

transfer rate
See *bandwidth*

transmission method
Refers to the electrical implementation of the SCSI specification for a bus.

tune
To modify the kernel by changing the values of kernel variables, which will improve system performance.

UBC
See *Unified Buffer Cache*

UBC LRU
The Unified Buffer Cache least-recently used (UBC LRU) pages are the oldest pages that are being used by the UBC.

Unified Buffer Cache
A portion of physical memory that is used to cache most-recently accessed file system data.

UltraSCSI
Refers to a storage configuration of devices (adapters or controllers) and disks that doubles the performance of SCSI-2 configurations. UltraSCSI (also called Fast-20) supports increased bandwidth and throughput, and can support extended cable distances.

virtual address space
The array of pages that an application can map into physical memory. Virtual address space is used for anonymous memory (memory used for stack, heap, or `malloc`) and for file-backed memory (memory used for program text or shared libraries).

virtual memory subsystem
A subsystem that uses a portion of physical memory, disk swap space, and daemons and algorithms to control the allocation of memory to processes and to the UBC.

VLDB
Refers to very-large database (VLDB) systems, which are VLM systems that use a large and complex storage configuration. The following is a typical VLM/VLDB system configuration:

- An SMP system with two or more high-speed CPUs
- More than 4 GB of physical memory
- Multiple high-performance host bus adapters
- RAID storage configuration for high performance and high availability

VLM
Refers to very-large memory (VLM) systems, which utilize 64-bit architecture, multiprocessing, and at least 2 GB of memory.

vnode
The kernel data structure for an open file.

wired list
Pages that are wired and cannot be reclaimed.

wired memory
Pages of memory that are wired and cannot be reclaimed by paging.

working set

The set of virtual addresses that are currently mapped to physical addresses. The working set is a subset of the resident set and represents a snapshot of the process' resident set.

workload

The total number of applications running on a system and the users utilizing a system at any one time under normal conditions.

World Wide Names (WWN)

A unique number assigned to a subsystem by the Institute of Electrical and Electronics Engineers (IEEE) and set by the manufacturer prior to shipping. The worldwide name assigned to a subsystem never changes. Fibre Channel devices have both a node name and a port name worldwide name, both of which are 64-bit numbers.

zero-filled-on-demand page fault

A page fault that occurs when a requested address is accessed for the first time.

zone

A logical subset of the Fibre Channel devices that re connected to the fabric.

zoning

Allows partitioning of resources for management and access control. It may provide efficient use of hardware resources by allowing one switch to serve multiple clusters or even mulitple operating systems. It entails splitting the fabric into zones, whre each zone is essentially a virtual faric.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Z