Tru64 UNIX

System Configuration and Tuning

Part Number: AA-RH9GA-TE

July 1999

Product Version: Tru64 UNIX Version 5.0 or higher

This manual decribes high-performance and high-availability configurations and provides recommendations for improving operating system performance.

Compaq Computer Corporation
Houston, Texas

Contents

7 Managing CPU Performance

8 Managing Disk Storage Performance

9 Managing File System Performance

10 Managing Network Performance

Tables

About This Manual

This manual contains information about configuring Compaq Tru64 UNIX (formerly DIGITAL UNIX) for high performance and high availability. This manual also describes how to tune systems to improve performance.

For Tru64 UNIX, it is recommended that you use the graphical user interface (GUI) for system administration. 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 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.
- *Section 4.1* describes the steps that you can follow to configure and tune high-performance and high-availability systems.
- Tru64 UNIX is able to satisfy page faults by reclaiming pages from the free page list. This improves virtual memory and Unified Buffer Cache (UBC) performance.
- The `vm_perf` data structure and other data structures require you to specify a processor number.

- You can specify swap devices by using the `vm` subsystem attribute `swapdevice`. See *Section 6.2* for more information.
- LSM provides RAID 5 and hot spare support. See *Section 8.4* for more information.
- LSM block-change logging (BCL) has been replaced with dirty-region logging (DRL), which can improve the recovery time of mirrored volumes after a system failure.
- AdvFS supports direct I/O, which can significantly improve disk I/O throughput for applications that read or write data only once and do not reuse the data. See *Section 9.3.4.7* for more information.
- Smooth sync functionality improves AdvFS asynchronous I/O performance by preventing I/O spikes caused by the update daemon, increasing the chance of a buffer cache hit, and improving the consolidation of I/O requests. See *Section 9.3.6.5* for more information.
- There are two new AdvFS subsystem attributes, `AdvfsMinFreeAccess` and `AdvfsMaxFreeAccess`, which control the allocation of AdvFS access structures. See *Section 9.3.6.3* for more information.
- It is no longer necessary to preallocate space for the AdvFS bitmap metadata table (BMT) or increase the number of pages by which the BMT extent size grows. The operating system performs this function automatically. See *Section 9.3.5.6* for more information.
- You can modify the UFS block size by using the `newfs` command. See *Section 9.4.1.1* for more information.

Organization

This manual consists of eleven chapters and a glossary:

Chapter 1 Introduces the terms and concepts related to performance and availability.

Chapter 2 Describes how to characterize your applications and users and choose a hardware and software configuration that will meet your needs.

Chapter 3 Describes how to monitor subsystems and identify and solve performance problems. The chapter also describes how to tune the kernel by modifying subsystem attributes.

Chapter 4 Describes the steps for configuring and tuning a high-performance and high-availability system. This chapter also provides tuning guidelines for Internet servers, large-memory systems, and NFS servers, and describes how to solve common performance problems.

Chapter 5 Describes how to tune operating system limits in order to provide applications and users with more system resources.

Chapter 6	Describes memory operation and how to manage virtual memory performance.
Chapter 7	Describes how to monitor CPUs and improve CPU performance.
Chapter 8	Describes how to manage the disk I/O subsystem, including the Logical Storage Manager (LSM) and hardware RAID subsystems.
Chapter 9	Describes how to manage file system performance.
Chapter 10	Describes how to monitor and tune the network subsystem.
Chapter 11	Describes how to improve application performance.
Glossary	Lists the terms relating to system performance and availability.

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*
- *Logical Storage Manager*
- *AdvFS Administration*
- *Systems & Options Catalog*

Icons on Tru64 UNIX Printed Books

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

G	Books for general users
S	Books for system and network administrators
P	Books for programmers

D	Books for device driver writers
R	Books for reference page users

Some books in the documentation help meet the needs of several audiences. For example, the information in some system books 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 books in the Tru64 UNIX documentation set.

Reader's Comments

Compaq 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`

- Mail:

  Compaq Computer Corporation
  UBPG Publications Manager
  ZKO3-3/Y32
  110 Spit Brook Road
  Nashua, NH 03062-2698

  A Reader's Comment form is located in the back of each printed manual. The form is postage paid if you mail it in the United States.

Please include the following information along with your comments:

- The full title of the book and the order number. (The order number is printed on the title page of this book and on its back cover.)
- 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 Compaq technical support office.

Information provided with the software media explains how to send problem reports to Compaq.

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.

1

Introduction to Performance and Availability

Businesses need a computing environment that is dependable and able to handle the workload placed on that environment. Users and applications place different demands on a system, and both require consistent performance with minimal down time. A system also must be able to absorb an increase in workload without a decline in performance.

By following the guidelines in this manual, you can configure and tune a dependable, high-performance Tru64 UNIX system that will meet your current and future computing needs.

This chapter provides the following information:

- Terminology that is used to define system performance (see Section 1.1)
- Introduction to high-performance configurations (see Section 1.2)
- Introduction to high-availability configurations (see Section 1.3)

Later chapters in this manual provide detailed information about choosing high-performance and high-availability configurations and improving performance.

1.1 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 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 than file system I/O, in some cases. 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.

1.2 High-Performance Configurations

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 in order 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 Compaq

Systems & Options Catalog and the Tru64 UNIX *Technical Overview* for information about hardware and operating system performance features.

1.2.1 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 1–1 describes the various hardware resources.

Table 1–1: 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 lookaside 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).

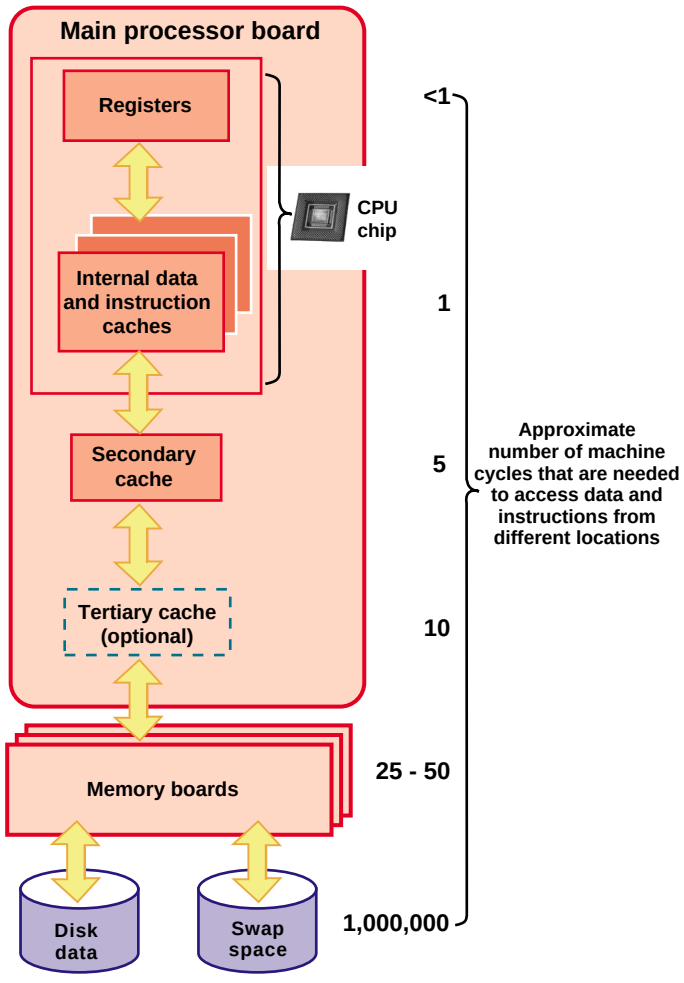
The hardware logic and the Privileged Architecture Library (PAL) code control much of the movement of addresses and data among the CPU cache, the secondary and tertiary caches, and physical memory. This movement is transparent to the operating system.

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



ZK-1362U-AI

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 Chapter 7 for more information.

1.2.2 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 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 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. 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

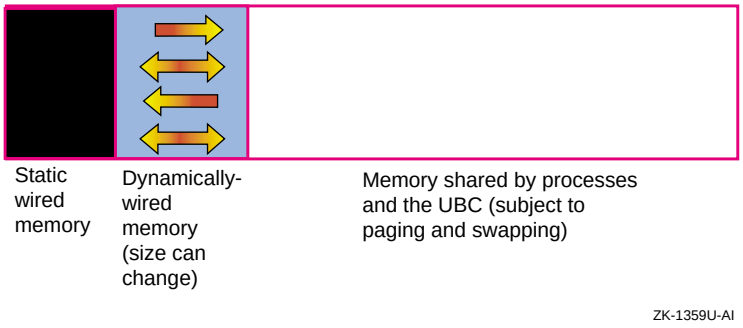
  Physical memory that is not wired is referred to as pageable memory. It is used for processes' most-recently accessed **anonymous memory** (modifiable virtual address space) and **file-backed memory** (memory that is used for program text or shared libraries). Pageable memory is also used to cache the 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 memory

(**swapping**). You can control virtual memory operation by tuning various vm subsystem attributes, as described in this chapter.

Figure 1–2 shows the division of physical memory.

Figure 1–2: Physical Memory Usage



Because retrieving data from memory is significantly faster than a disk I/O operation, **buffer caching** is used to cache recently-used disk data in physical memory. I/O performance is improved if the cached data is later reused.

There are several buffer caches that are used to cache data:

- The UBC caches the most-recently accessed file data for reads and writes and for page faults from mapped file regions, in addition to Advanced File System (AdvFS) metadata and file data. The UBC and processes compete for the portion of physical memory that is not wired by the kernel.
- The **AdvFS buffer cache** is a subset of the UBC.
- The **metadata buffer cache** caches only UFS and CDFS metadata and is part of static wired memory.

Various kernel subsystem attributes control the amount of memory available to processes and to the file system buffer caches, and the rate of page reclamation. You may be able to tune the attributes in order to optimize memory performance. See Chapter 6 for more information.

1.2.3 Disk Storage

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, suitable for different environments.

Host bus adapters, RAID controllers, and disks have various performance features and support different parallel Small Computer System Interface (**SCSI**) variants. SCSI is a device and interconnect technology that continues to evolve in terms of high performance, availability, and configuration flexibility. See Section 1.2.3.1 for more information about RAID functionality. See Section 1.2.3.2 for more information about SCSI.

See Chapter 2 and Chapter 8 for more information about storage configurations.

1.2.3.1 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 LSM or a hardware-based RAID subsystem.

There are four primary RAID levels:

- **RAID 0**—Also known as data or disk **striping**, RAID 0 divides data into blocks and distributes the blocks across multiple disks in a array, which improves throughput. Striping does not provide disk data availability.
- **RAID 1**—Also known as data or disk **mirroring**, RAID 1 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 RAID 1 with RAID 0 in order to mirror striped data or disks.
- **RAID 3**—A type of **parity RAID**, RAID 3 divides data blocks and distributes the data across a disk array, providing parallel access to data and increasing bandwidth. RAID 3 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.
- **RAID 5**—A type of parity RAID, RAID 5 distributes data blocks across disks in an array. RAID 5 allows independent access to data and can handle simultaneous I/O operations, which improves throughput. RAID 5 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 RAID 3/5**), which combines the benefits of RAID 3 and RAID 5 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 (one or more disks have failed), and recovery (subsystem is recovering from failure).

Table 1–2 compares the performance features and degrees of availability for the different RAID levels.

Table 1–2: RAID Level Performance and Availability Comparison

RAID Level	Performance Feature	Degree of Availability
RAID 0 (striping)	Balances I/O load and improves throughput	Lower than single disk
RAID 1 (mirroring)	Improves read performance, but degrades write performance	Highest
RAID 0+1	Balances I/O load and improves throughput, but degrades write performance	Highest
RAID 3	Improves bandwidth, but performance may degrade if multiple disks fail	Higher than single disk
RAID 5	Improves throughput, but performance may degrade if multiple disks fail	Higher than single disk
Dynamic parity RAID	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 RAID 1 (mirroring), because mirroring *n* disks requires 2*n* 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 2 and Chapter 8 for more information about RAID configurations.

1.2.3.2 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), transmission method (single-ended or differential), and bus speed (Slow, Fast, or Ultra). 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.

1.2.3.2.1 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 and Fast SCSI, and a 32-bit data path for UltraSCSI. 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 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) to connect devices with different data paths.

1.2.3.2.2 Transmission Methods

The **transmission method** for a bus refers to the electrical implementation of the SCSI specification. **Single-ended SCSI** is a low-cost solution for devices that are usually located within the same cabinet. Single-ended SCSI usually requires short cable lengths. **Differential SCSI** can be used to connect devices that are up to 25 meters apart.

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.

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 can directly connect devices only if they have the same transmission method (differential or single-ended) and data path (narrow or wide). Use a SCSI signal converter or an UltraSCSI extender to connect devices with different transmission methods or data paths.

1.2.3.2.3 SCSI Bus Speeds

The **SCSI bus speed**, also called the transfer rate or bandwidth, is the number of transfers per second. Fast 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 Compaq *Systems & Options Catalog* for information about SCSI device support.

Table 1–3 shows the currently available bus speeds.

Table 1–3: 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)
Slow	5	5	10
Fast SCSI	10	10	20
UltraSCSI	20	20	40

Fast SCSI bus speed, 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.

UltraSCSI devices are wide and can be either single-ended or differential. All UltraSCSI components are backward compatible with regular SCSI-2 components.

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 couples 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).

In addition to extending the effective length of a bus, UltraSCSI bus extenders can be used as **SCSI signal converters**, so you can connect differential bus segments to single-ended bus segments. This allows you to

mix differential and single-ended devices on the same bus. A bus extender also enables bus segments to be isolated from each other for maintenance reasons.

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.

1.2.3.2.4 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 1–4 shows the maximum bus lengths for different bus speeds and transmission methods.

Table 1–4: 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 1–4: 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.

1.2.4 Network Subsystem

The operating system supports 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 availability.

Kernel attributes control network subsystem operation. You may be able to optimize network performance for your applications and workload by modifying kernel subsystem attribute values.

1.3 High-Availability Configurations

In addition to high performance, many environments require some degree of **high availability**, which is the ability to withstand a hardware or software failure. Resources (for example, disk data, systems, and network connections) can be made highly available by using some form of resource duplication or **redundancy**. In some cases, an automatic **failover** mechanism may also be used in order to make the resource failure virtually imperceptible to users.

There are various degrees of availability, and you must determine how much you need for your environment. Critical operations may require a configuration that does not have a **single point of failure**; that is, a configuration in which you have duplicated each vital resource. However, some environments may be able to accommodate some down time and may require only data to be highly available.

Figure 1–3 shows a configuration that is vulnerable to various single failures, including network, disk, and bus failures.

Figure 1–3: Configuration with Potential Points of Failure

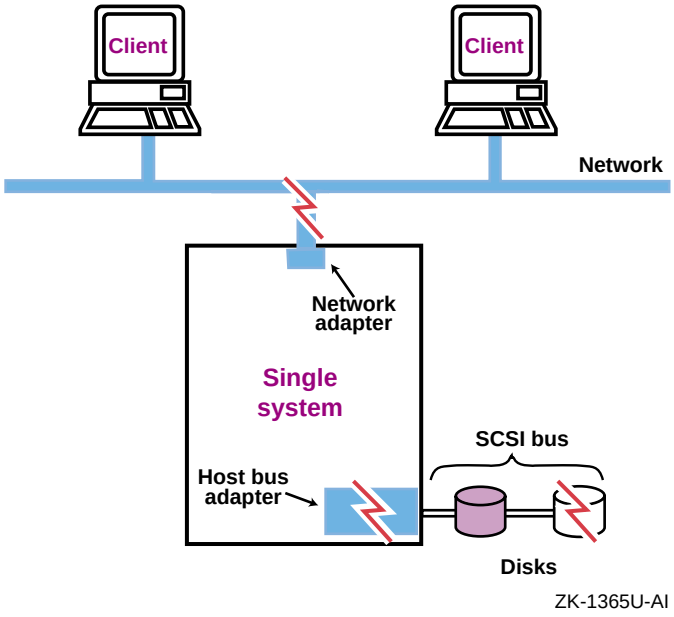
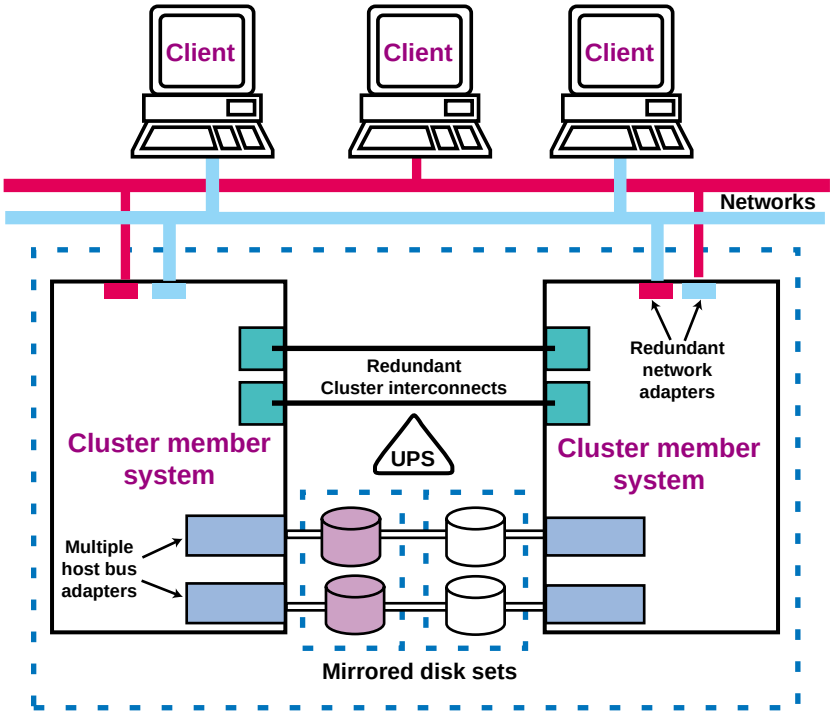


Figure 1–4 shows a fully redundant cluster configuration with no single point of failure.

Figure 1–4: Fully Redundant Cluster Configuration



By duplicating important resources, a configuration can be resistant to resource failures, as follows:

- Disk data

  RAID technology provides you with various degrees of data availability. For example, you can use RAID 1 (mirroring) to replicate data on different disks. If one disk fails, a copy is still available to users and applications. You can also use parity RAID for high data availability. The parity information is used to reconstruct data if a failure occurs.

  In addition, to protect data against a host bus adapter or bus failure, mirror data the across disks located on different buses.

- Network access

  You can also 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.

In addition, you can use **NetRAIN** (redundant array of independent network adapters) to provide highly-available network access. NetRAIN enables you to configure multiple interfaces on the same LAN segment into a single interface, and provides failover support for network adapter and network connections.

- System or application

  To make systems and applications highly available, you must use a TruCluster Server to set up a **cluster**, which is a loosely coupled group of servers configured as **member systems** and usually connected to shared disk storage. Software applications are installed on every member system, but only one system runs an application at one time and makes it available to users.

  A cluster utilizes a failover mechanism to protect against failures. If a member system fails, all cluster-configured applications running on that system will fail over to a viable member system. The new system then starts the applications and makes them available to users.

  Some cluster products support a high-performance **cluster interconnect** that provides fast and reliable communications between members. You can configure redundant cluster interconnects for high availability. If one cluster interconnect fails, the cluster members can still communicate over the remaining interconnect.

- Power

  Systems and storage units are vulnerable to power failures. To protect against a power supply failure, use redundant power supplies. You can also protect disks against a power supply failure in a storage cabinet by mirroring across independently powered cabinets.

  In addition, use an uninterruptible power supply (UPS) system to protect against a total power failure (for example, the power in a building fails). A UPS system depends on a viable battery source and monitoring software.

You must repair or replace a failed component as soon as possible to maintain some form of redundancy. This will help to ensure that you do not experience down time.

Production environments often require that resources be resistant to multiple failures. The more levels of resource redundancy, the greater the resource availability. For example, if you have only two cluster member systems and one fails, the remaining system is now a potential point of failure. Therefore, a cluster with three or more member systems has higher availability than a two-system cluster, because it has more levels of redundancy and can survive multiple system failures.

Availability is also measured by a resource's **reliability**, which is the average amount of time that a component will perform before a failure that causes a loss of data. It is often expressed as the mean time to data loss (MTDL), the mean time to first failure (MTTF), and the mean time between failures (MTBF).

See Section 2.6 for detailed information about setting up a high-availability configuration.

2

Planning a High-Performance and High-Availability Configuration

A high-performance configuration is one that will rapidly respond to the demands of a normal workload, and also handle an increase in the workload. A high-availability configuration provides protection against single points of failure.

This chapter describes how to perform the following tasks:

- Identify a resource model for your workload (Section 2.1)
- Identify performance and availability goals (Section 2.2)
- Choose high-performance system hardware (Section 2.3)
- Choose high-performance disk storage hardware (Section 2.4)
- Choose how to manage your disk storage (Section 2.5)
- Choose a high-availability configuration (Section 2.6)

2.1 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 a application performs many disk write operations, you may not want to choose a RAID 1 (mirrored) configuration.

Use Table 2–1 to help you determine the resource model for your workload and identify a possible configuration solution for each model.

Table 2–1: 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	High-performance SCSI adapters, striping, RAID 5, or dynamic parity RAID
Requires high bandwidth	High-performance adapters, wide devices, RAID 3, or dynamic parity RAID
Performs many large sequential data transfers	High-performance disks, wide devices, striping, parity RAID
Performs many small data transfers	RAID 5
Issues predominantly read transfers	Mirroring, RAID 5, 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

2.2 Identifying Performance and Availability Goals

Before you choose a configuration, you must determine the level of performance and availability that you need. In addition, you must account for cost factors and plan for future workload expansion.

When choosing a system and disk storage configuration, you must be sure to evaluate the configuration choices in terms of the following criteria:

- Performance

  You must determine an acceptable level of performance for the applications and users. For example, you may want a real-time environment that responds immediately to user input, or you may want an environment that has high throughput.

- Availability

  You must determine how much availability is needed. Some environments require only highly available data. Other environments require you to eliminate all single points of failure.

- Cost

  You must determine the cost limitations for the environment. For example, solid-state disks provide high throughput and high bandwidth, but at a high cost.

- Scalability

  A system that is scalable is able to utilize additional resources (for example, additional CPUs) with a predictable increase in performance. Scalability can also refer to the ability of a system to absorb an increase in workload without a significant performance degradation. Be sure to include in your plans any potential workload increases and, if necessary, choose a configuration that is scalable.

After you determine the goals for your environment, you can choose the system and disk storage configuration that will address these goals.

2.3 Choosing System Hardware

Different systems provide different configuration and performance features. A primary consideration for choosing a system is its CPU and memory capabilities. Some systems support multiple CPUs, fast CPU speeds, and very-large memory (VLM) configurations.

Because very-large database (VLDB) systems and cluster systems usually require many external I/O buses, another consideration is the number of I/O bus slots in the system. Some systems support I/O expansion units for additional I/O capacity.

Be sure that a system is adequately scalable, which will determine whether you can increase system performance by adding resources, such as CPU and memory boards. If applicable, choose a system that supports RAID controllers, high-performance network adapters, and cluster products.

Table 2–2 describes some hardware options that can be used in a high-performance system, and the performance benefit for each option.

Table 2–2: High-Performance System Hardware Options

Hardware Option	Performance Benefit
Multiple CPUs (Section 2.3.1)	Improves processing time
Fast CPU speed (Section 2.3.1)	Improves processing time

Table 2–2: High-Performance System Hardware Options (cont.)

Hardware Option	Performance Benefit
Onboard CPU cache (Section 2.3.1)	Improves processing time
Very-large memory (Section 2.3.2)	Improves processing time and decreases disk I/O latency
Large I/O capacity (Section 2.3.3)	Allows you to connect many I/O adapters and controllers for disk storage and network connections
Support for high-performance disk storage (Section 2.3.4)	Improves overall system performance and availability
Support for high-performance networks (Section 2.3.5)	Increases network access and network performance

For detailed information about hardware performance features, see the Compaq *Systems & Options Catalog*.

The following sections describe these hardware options in detail.

2.3.1 CPU Configuration

To choose a CPU configuration that will meet your needs, you must determine your requirements for the following:

- Number of CPUs

  Only certain types of systems support multiprocessing. If your environment is CPU-intensive or if your applications can benefit from multiprocessing, you may want a system that supports multiple CPUs.

  Depending on the type of multiprocessing system, you can install two or more CPUs. You must determine the number of CPUs that you need, and then choose a system that supports that number of CPUs and has enough backplane slots available for the CPU boards.

- CPU processing speed

  CPUs have different processing speeds and other performance features. For example, some processors support EV56 technology.

  If your environment is CPU-intensive, you may want to choose a system that supports high-performance CPUs.

- CPU cache size

  CPUs have different sizes for on-chip caches, which can improve performance. Some systems have secondary caches that reside on the main processor board, and some have tertiary caches.

  Caches that reside on the CPU chip can vary in size up to a maximum of 64 KB (depending on the type of processor). These caches include the

translation lookaside buffer, the high-speed internal virtual-to-physical translation cache, the high-speed internal instruction cache, and the high-speed internal data 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. The tertiary cache is not available on all Alpha CPUs; otherwise, it is identical to the secondary cache.

2.3.2 Memory and Swap Space Configuration

You must determine the total amount of memory and swap space that you need to handle your workload. Insufficient memory resources and swap space will cause performance problems. In addition, your memory bank configuration will affect performance.

To configure memory and swap space, perform the following tasks:

1. Determine how much physical memory your workload requires and choose a system that provides the necessary memory and has enough backplane slots for memory boards (Section 2.3.2.1).
2. Choose a swap space allocation mode (Section 2.3.2.2).
3. Determine how much swap space you need (Section 2.3.2.3).
4. Configure swap space in order to efficiently distribute the disk I/O (Section 6.2).

The following sections describe these tasks.

2.3.2.1 Determining Your Physical Memory Requirements

You must have enough system memory to provide an acceptable level of user and application performance. The amount of memory installed in your system must be at least as much as the sum of the following:

- The total amount of memory that will be wired, including operating system data and text, system tables, the metadata buffer cache, and dynamically allocated data structures
- The total amount of memory your processes need for anonymous memory, which holds data elements and structures that are modified during process execution (heap space, stack space, and space allocated by the `malloc` function)
- The total amount of memory that the UBC requires to cache data

In addition, each network connection to your server requires the following memory resources:

- Kernel socket structure
- Internet protocol control block (`inpcb`) structure
- TCP control block structure
- Any additional 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), so you need 10 MB of memory to accommodate 10,000 connections. There is no limit on a system's ability to handle millions of TCP connections, if you have enough memory resources to service the connections. However, when memory is low, the server will reject new connection requests until enough existing connections are freed. Use the `netstat -m` command to display the memory that is currently being used by the network subsystem.

To ensure that your server has enough memory to handle high peak loads, you should have available 10 times the memory that is needed on a busy day. For optimal performance and for scalability, configure more than the minimum amount of memory needed.

2.3.2.2 Choosing a Swap Space Allocation Mode

There are two modes that you can use to allocate swap space. The modes differ in how the virtual memory subsystem reserves swap space for anonymous memory (modifiable virtual address space). Anonymous memory is memory that is not backed by a file, but is backed by swap space (for example, stack space, heap space, and memory allocated by the `malloc` function). 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).

You specify the swap space allocation mode by using the `vm` subsystem attribute `vm_swap_eager`. Specify 1 to enable immediate mode (the default); specify 0 to enable deferred mode.

See the *System Administration* manual for more information on swap space allocation methods.

2.3.2.3 Determining Swap Space Requirements

Swap space is used to hold the recently accessed modified pages from processes and from the UBC. In addition, if a crash dump occurs, the operating system writes all or part of physical memory to swap space.

It is important to configure a sufficient amount of swap space and to distribute swap space across multiple disks. An insufficient amount of swap space can severely degrade performance and prevent processes from running or completing. A minimum of 128 MB of swap space is recommended.

The optimal amount of swap space for your configuration depends on the following factors:

- Total amount of memory configured in the system—In general, systems with large amounts of memory require less swap space than low-memory systems.
- Your applications' anonymous memory requirements—Anonymous memory holds data elements and structures that are modified during process execution, such as heap space, stack space, and memory allocated with the `malloc` function.
- Crash dump configuration—Swap space should be large enough to hold a complete crash dump file. Tru64 UNIX supports compressed crash dumps, which require less swap space to hold the dump file than uncompressed crash dumps. See the *System Administration* manual for more information on compressed crash dumps.

- Swap space allocation mode—Less swap space is required if you are using deferred mode instead of immediate mode.
- Whether you use the AdvFS `verify` command—If you want to use the AdvFS `verify` command, allocate an additional 5 percent of total memory for swap space.

To calculate the amount of swap space required by your configuration, follow these steps:

1. Determine the total amount of anonymous memory (modifiable virtual address space) required by all of your processes. To do this, invoke the `ps -o vsz` command and add the values in the `VSZ` (virtual address size) column. For example:

```
# ps -o vsz
 VSZ
 536K
2.16M
2.05M
2.16M
2.16M
3.74M
2.16M
#
```

2. If you are using immediate mode, add 10 percent to the total amount of anonymous memory.

   If you are using deferred mode, divide the total amount of anonymous memory by 2.

3. If you are using AdvFS, add 5 percent of total memory to this value. The resulting value represents the optimal amount of swap space.

You can configure swap space when you first install the operating system, or you can add swap space at a later date. In addition, swap space must following the guidelines for distributing disk I/O. See Section 6.2 for information about adding swap space after installation and configuring swap space for high performance.

2.3.3 I/O Bus Slot Capacity

Systems provide support for different numbers of I/O bus slots to which you can connect external storage devices and network adapters. Some enterprise systems provide up to 132 PCI slots for external storage. These systems are often used in VLDB systems and cluster configurations.

You must ensure that the system you choose has sufficient I/O buses and slots available for your disk storage and network configuration. Some systems support I/O expansion units for additional I/O capacity.

2.3.4 Support for High-Performance Disk Storage

Systems support different local disk storage configurations, including multiple storage shelves, large disk capacity, and UltraSCSI devices. In addition, some systems support high-performance PCI buses, which are required for hardware RAID subsystems and clusters.

You must ensure that the system you choose supports the disk storage configuration that you need. See Section 2.4 and Section 2.5 for more information on disk storage configurations.

2.3.5 Support for High-Performance Network

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 availability.

2.4 Choosing Disk Storage Hardware

The disk storage subsystem is used for both data storage and for swap space. Therefore, an incorrectly configured or tuned disk subsystem can degrade both disk I/O and virtual memory performance. Using your resource model, as described in Section 2.1, choose the disk storage hardware that will meet your performance needs.

Table 2–3 describes some hardware options that can be used in a high-performance disk storage configuration and the performance benefit for each option.

Table 2–3: High-Performance Disk Storage Hardware Options

Hardware Option	Performance Benefit
Fast disks (Section 2.4.1)	Improves disk access time and sequential data transfer performance
Solid-state disks (Section 2.4.2)	Provides very low disk access time
Wide devices (Section 2.4.3)	Provides high bandwidth and improves performance for large data transfers
High-performance host bus adapters (Section 2.4.4)	Increases bandwidth and throughput, and supports wide data paths and fast bus speeds
DMA host bus adapters (Section 2.4.5)	Relieves CPU of data transfer overhead

Table 2–3: High-Performance Disk Storage Hardware Options (cont.)

Hardware Option	Performance Benefit
RAID controllers (Section 2.4.6 and Section 8.5)	Decreases CPU overhead; increases the number of disks that can be connected to an I/O bus; provides RAID functionality; and supports write-back caches
Fibre Channel (Section 2.4.7)	Provides high access speeds and other high-performance features
Prestoserve (Section 2.4.8)	Improves synchronous write performance

For detailed information about hardware performance features, see the Compaq *Systems & Options Catalog.*

The following sections describe some of these high-performance disk storage hardware options in detail.

2.4.1 Fast Disks

Disks that spin with a high rate of revolutions per minute (RPM) have a low disk access time (latency). High-RPM disks are especially beneficial to the performance of sequential data transfers.

High-performance disks (7200 RPM) can improve performance for many transaction processing applications (TPAs). UltraSCSI disks (10,000 RPM) are ideal for demanding applications, including network file servers and Internet servers, that require high bandwidth and high throughput.

2.4.2 Solid-State Disks

Solid-state disks provide outstanding performance compared to magnetic disks, but at a higher cost. By eliminating the seek and rotational latencies that are inherent in magnetic disks, solid-state disks can provide very high disk I/O performance. Disk access time is under 100 microseconds, which allows you to access critical data more than 100 times faster than with magnetic disks.

Available in both wide (16-bit) and narrow (8-bit) versions, solid-state disks are ideal for response-time critical applications with high data transfer rates, such as online transaction processing (OLTP), and applications that require high bandwidth, such as video applications.

Solid-state disks complement hardware RAID configurations by eliminating bottlenecks caused by random workloads and small data sets. Solid-state disks also provide data reliability through a nonvolatile data-retention system.

For the best performance, use solid-state disks for your most frequently accessed data to reduce the I/O wait time and CPU idle time. In addition, connect the disks to a dedicated bus and use a high-performance host bus adapter.

2.4.3 Devices with Wide Data Paths

Disks, host bus adapters, SCSI controllers, and storage expansion units support wide data paths, which provide nearly twice the bandwidth of narrow data paths. Wide devices can greatly improve I/O performance for large data transfers.

Disks with wide (16-bit) data paths provide twice the bandwidth of disks with narrow (8-bit) data paths. To obtain the performance benefit of wide disks, all the disks on a SCSI bus must be wide. If you use both wide and narrow disks on the same SCSI bus, the bus performance will be constrained by the narrow disks.

2.4.4 High-Performance Host Bus Adapters

Host bus adapters and interconnects provide different performance features at various costs. For example, FWD (fast, wide, and differential) SCSI bus adapters provide high bandwidth and high throughput connections to disk devices. Other adapters support UltraSCSI.

In addition, some host bus adapters provide dual-port (dual-channel) support, which allows you to connect two buses to one I/O bus slot.

Bus speed (the rate of data transfers) depends on the host bus adapter. Different adapters support bus speeds ranging from 5 million bytes per second (5 MHz) for slow speed to 40 million bytes per second (20 MHz) for UltraSCSI.

You must use high-performance host bus adapters, such as the KZPSA adapter, to connect systems to high-performance RAID array controllers.

2.4.5 DMA Host Bus Adapters

Some host bus adapters support direct memory access (DMA), which enables an adapter to bypass the CPU and go directly to memory to access and transfer data. For example, the KZPAA is a DMA adapter that provides a low-cost connection to SCSI disk devices.

2.4.6 RAID Controllers

RAID controllers are used in hardware RAID subsystems, which greatly expand the number of disks connected to a single I/O bus, relieve the

CPU of the disk I/O overhead, and provide RAID functionality and other high-performance and high-availability features.

There are various types of RAID controllers, which provide different features. High-performance RAID array controllers support dynamic parity RAID and battery-backed, write-back caches. Backplane RAID storage controllers provide a low-cost RAID solution.

See Section 8.5 for more information about hardware RAID subsystems.

2.4.7 Fibre Channel

Fibre Channel is a high-performance I/O bus that is an example of serial SCSI and provides network storage capabilities. Fibre Channel supports multiple protocols, including SCSI, Intellitan Protocol Interface (IPI), TCP/IP, and High-Performance Peripheral Interface (HPPI).

Fibre Channel is based on a network of intelligent switches. Link speeds are available up to 100 MB/sec full duplex. Although Fibre Channel is more expensive than parallel SCSI, Fibre Channel Arbitrated Loop (FC-AL) decreases costs by eliminating the Fibre Channel fabric and using connected nodes in a loop topology with simplex links. In addition, an FC-AL loop can connect to a Fibre Channel fabric.

2.4.8 Prestoserve

The Prestoserve product is a combination of NVRAM hardware and software. Prestoserve can speed up synchronous disk writes, including NFS server access, by reducing the amount of disk I/O.

Prestoserve uses nonvolatile, battery-backed memory to temporarily cache file system writes that otherwise would have to be written to disk. This capability improves performance for systems that perform large numbers of synchronous writes.

To optimize Prestoserve cache use, you may want to enable Prestoserve only on the most frequently used file systems, or configure Prestoserve to cache only UFS or AdvFS metadata. In addition, Prestoserve can greatly improve performance for NFS servers. See Section 9.2.9 for more information.

You cannot use Prestoserve in a cluster or for nonfile system I/O.

2.5 Choosing How to Manage Disks

There are various methods that you can use to manage your disk storage configuration. These methods provide different performance and availability features. You must understand your workload resource model, as described in Section 2.1, to determine the best way to manage disk storage.

In addition, if you require highly-available disk storage, see Section 2.6 for information about the available options.

After you choose a method for managing disk storage, see Chapter 8 for configuration guidelines.

Table 2–4 describes some options for managing disk storage and the performance benefit and availability impact for each option.

Table 2–4: High-Performance Disk Storage Configuration Solutions

Configuration Option	Performance Benefit
Shared pool of storage (LSM) (Section 2.5.1)	Facilitates management of large amounts of storage (LSM also provides RAID functionality)
Data or disk striping (RAID 0) (Section 2.5.2)	Distributes disk I/O and improves throughput, but decreases availability
RAID 3 (Section 2.5.3)	Improves bandwidth and provides availability
RAID 5 (Section 2.5.3)	Improves throughput and provides availability
Dynamic parity RAID (Section 2.5.3)	Improves overall disk I/O performance and provides availability
Mirroring (RAID 1) (Section 2.6.2)	Improves read performance and provides high availability, but decreases write performance
Mirroring striped data or disks (Section 2.5.2 and Section 2.6.2)	Combines the performance benefits of RAID 0 with the availability benefits of RAID 1

The following sections describe some of these high-performance storage configurations in detail.

2.5.1 Using a Shared Pool of Storage for Flexible Management

There are two methods that you can use to manage the physical disks in your environment. The traditional method of managing disks and files is to divide each disk into logical areas called disk partitions, and to then create a file system on a partition or use a partition for raw I/O.

Each disk type has a default partition scheme. The `disktab` database file lists the default disk partition sizes. The size of a partition determines the amount of data it can hold. It can be time-consuming to modify the size of a partition. You must back up any data in the partition, change the size by using the `disklabel` command, and then restore the data to the resized partition.

An alternative to managing disks with static disk partitions is to use the Logical Storage Manager (LSM) to set up a shared pool of storage that consists of multiple disks. You can create virtual disks (LSM volumes) from this pool of storage, according to your performance and capacity needs, and then place file systems on the volumes or use them for raw I/O.

LSM provides you with flexible and easy management for large storage configurations. Because there is no direct correlation between a virtual disk and a physical disk, file system or raw I/O can span disks as needed. In addition, you can easily add disks to and remove disks from the pool, balance the load, and perform other storage management tasks.

LSM also provides you with high-performance and high-availability RAID functionality, hot spare support, and load balancing.

See Section 8.4 for information about LSM configurations.

2.5.2 Striping Data or Disks to Distribute I/O

Data or disk striping (RAID 0) distributes disk I/O and can improve throughput. The striped data is divided into blocks (sometimes called chunks or stripes) and distributed across multiple disks in a array. Striping enables parallel I/O streams to operate concurrently on different devices, so that I/O operations can be handled simultaneously by multiple devices.

LSM enables you to stripe data across different buses and adapters. Hardware RAID subsystems provide only disk striping.

LSM data striping provides better performance than disk striping with hardware RAID and supports more flexible configurations. With LSM, you can create a striped 36-GB volume comprised of three 9-GB disks and two 4.5-GB disks, or you can create a striped 13.5-GB volume comprised of two 4.5-GB disks and half of a 9-GB disk. The other half of the 9-GB disk can be used in other LSM volumes.

The performance benefit of striping depends on the size of the stripe and how your users and applications perform disk I/O. For example, if an application performs multiple simultaneous I/O operations, you can specify a stripe size that will enable each disk in the array to handle a separate I/O operation. If an application performs large sequential data transfers, you can specify a stripe size that will distribute a large I/O evenly across the disks.

For volumes that receive only one I/O at a time, you may not want to use striping if access time is the most important factor. In addition, striping may degrade the performance of small data transfers, because of the latencies of the disks and the overhead associated with dividing a small amount of data.

Striping decreases data availability because one disk failure makes the entire disk array unavailable. To make striped data or disks highly available, you can combine RAID 0 with RAID 1 to mirror the striped data or disks.

See Chapter 8 for more information about LSM and hardware RAID subsystems.

2.5.3 Using Parity RAID to Improve Disk Performance

Parity RAID provides both high performance and high availability. Tru64 UNIX supports three types of parity RAID, each with different performance and availability benefits:

- RAID 3—Divides data blocks and distributes the data across a disk array, providing parallel access. RAID 3 provides a high data transfer rate and increases bandwidth, but it provides no improvement in throughput (the I/O transaction rate).

  Use RAID 3 to improve the I/O performance of applications that transfer large amounts of sequential data. RAID 3 provides no improvement for applications that perform multiple I/O operations involving small amounts of data.

  RAID 3 also provides high data availability by storing redundant parity information on a separate disk. The parity information is used to regenerate data if a disk in the array fails. However, performance degrades as multiple disks fail, and data reconstruction is slower than if you had used mirroring.

- RAID 5—Distributes data blocks across disks in an array. RAID 5 allows independent access to data and can handle simultaneous I/O operations.

  RAID 5 can be used for configurations that are mainly read-intensive. RAID 5 is not suitable for write-intensive applications.

  As a cost-efficient alternative to mirroring, you can use RAID 5 to improve the availability of rarely-accessed data. RAID 5 provides high data availability by distributing redundant parity information across disks. Each array member contains enough parity information to regenerate data if a disk fails. However, performance may degrade and data may be lost if multiple disks fail. In addition, data reconstruction is slower than if you had used mirroring. LSM supports only RAID 5 for parity RAID.

- Dynamic parity RAID—Dynamically adjusts, according to the workload, between data transfer-intensive algorithms and I/O operation-intensive algorithms, combining the performance benefits of RAID 3 and RAID 5. Also known as adaptive RAID 3/5, dynamic parity RAID improves disk I/O performance for a wide variety of applications. Only high-performance RAID controllers support dynamic parity RAID.

LSM can provide more flexible RAID configurations than hardware RAID, because LSM is able to utilize portions of disks, instead of requiring complete disks. See Chapter 8 for more information about LSM and hardware RAID subsystems.

2.6 Choosing a High-Availability Configuration

You can set up a configuration that provides the level of availability that you need. For example, you can make only disk data highly available or you can set up a cluster configuration with no single point of failure, as shown in Figure 1–4.

Table 2–5 lists each possible point of failure, the configuration solution that will provide high availability, and any performance benefits and tradeoffs.

Table 2–5: High-Availability Configurations

Point of Failure	Configuration Solution	Benefits and Tradeoffs
Single system	Latest hardware, firmware, and operating system releases	Provides the latest hardware and software enhancements, but may require down time during upgrade
	Cluster with at least two systems (Section 2.6.1)	Improves overall performance by spreading workload across member systems, but increases costs and management complexity
Multiple systems	Cluster with more than two members (Section 2.6.1)	Improves overall performance by spreading workload across member systems, but increases costs and management complexity
Cluster interconnect	Second cluster interconnect (Section 2.6.1)	Increases costs
Disk	Mirrored data or disks (Section 2.6.2)	Improves read performance, but increases costs and decreases write performance
	Parity RAID (Section 2.6.2)	Improves disk I/O performance, but increases management complexity and decreases performance when under heavy write loads and in failure mode
Host bus adapter or bus	Mirrored data across disks on different buses (Section 2.6.2)	Improves read performance, but increases costs and decreases write performance

Table 2–5: High-Availability Configurations (cont.)

Point of Failure	Configuration Solution	Benefits and Tradeoffs
Network connection	Multiple network connections or use NetRAIN (Section 2.6.3)	Improves network access and possibly performance, but increases costs
System cabinet power supply	Redundant power supplies (Section 2.6.4)	Increases costs
Storage unit power supply	Redundant power supplies or mirroring across cabinets with independent power supplies (Section 2.6.4 and Section 2.6.2)	Increases costs
Total power supply	Battery-backed uninterruptible power supply (UPS) system (Section 2.6.4)	Increases costs

The following sections describe some of the previous high-availability configurations in detail.

2.6.1 Using a Cluster for System Availability

If users and applications depend on the availability of a single system for CPU, memory, data, and network resources, they will experience down time if a system crashes or an application fails. To make systems and applications highly available, you must use the TruCluster Server product to set up a cluster.

A cluster is a loosely coupled group of servers configured as member systems and connected to highly available shared disk storage and common networks. Software applications are installed on every member system, but only one system runs an application at one time.

If a resource for a member system fails (for example, a network adapter or a bus fails), all cluster-configured applications running on that system will fail over to a viable member system. The new system then starts the applications and makes them available to users.

To protect against multiple system failures, use more than two member systems in a cluster.

In addition, TruCluster Server supports a high-performance cluster interconnect that enables fast and reliable communications between members. To protect against interconnect failure, use redundant cluster interconnects.

You can use only specific systems, host bus adapters, RAID controllers, and disks with the cluster products. In addition, member systems must have enough I/O bus slots for adapters, controllers, and interconnects.

You can use LSM or hardware RAID to improve the performance and availability of a cluster's shared storage. For example, LSM enables you to create a mirrored and striped volume that can be used to access an LSM volume simultaneously from different cluster member systems.

See the TruCluster Server *Software Product Description* for detailed information about the product.

2.6.2 Using RAID for Disk Data Availability

RAID technology provides you with high data availability, in addition to high performance. RAID 1 (data or disk mirroring) provides high data availability by maintaining identical copies of data on different disks in an array. If the original disk fails, the copy is still available to users and applications. To protect data against a host bus adapter or bus failure, mirror across disks located on different buses.

Mirroring can improve read performance because data can be read from two different locations. However, it decreases disk write performance, because data must be written to two different locations.

LSM provides more flexible RAID configurations than hardware RAID. LSM mirrors (and stripes) data. For example, LSM can mirror (and stripe) data across portions of disks or across different types of disks. Hardware RAID subsystems mirror entire disks.

Hardware RAID subsystems and LSM also use parity RAID, to provide high data availability and high performance. With parity RAID, data is spread across disks, and parity information is used to reconstruct data if a failure occurs. Tru64 UNIX supports three types of parity RAID (RAID 3, RAID 5, and dynamic parity RAID) and each provides different performance and availability benefits. LSM supports only RAID 5. Only high-performance RAID controllers support dynamic parity RAID. See Section 2.5.3 for more information.

See Chapter 8 for more information about LSM and hardware RAID subsystems.

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

2.6.4 Using Redundant Power Supplies and Systems

To protect against a power supply failure for cabinets, systems, and storage shelves, use redundant power supplies. Alternately, for disk storage units, you can mirror across cabinets with different power supplies.

In addition, use an uninterruptible power supply (UPS) system to protect against a total power failure (for example, the power in a building fails). A UPS system depends on a viable battery source and monitoring software.

3

Monitoring Systems and Diagnosing Performance Problems

You must gather a wide variety of performance information in order 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.

This chapter describes how to perform the following tasks:

- Understand how the system logs event messages (Section 3.1)
- Set up system accounting and disk quotas to track and control resource utilization (Section 3.2)
- Establish a method to continuously monitor system performance (Section 3.3)
- Use tools to gather a variety of performance information (Section 3.4)
- Profile and debug kernels (Section 3.5)
- Access and modify kernel subsystem attributes (Section 3.6)

After you identify a performance problem or an area in which performance is deficient, you can identify an appropriate solution. See Chapter 4 for information about improving system performance.

3.1 Obtaining Information About System Events

It is recommended that you set up a routine to continuously monitor system events and to 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. You should 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 3.1.1 for more information about EVM.
- 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 can analyze system events and provides a notification mechanism (for example, mail) that can warn of potential problems. See Section 3.1.2 for more information about DECevent.
- You can use the dia command or the uerf command to translate binary log files to ASCII format. See the *System Administration* manual, dia(8), and uerf(8) for information.

In addition, it is recommended 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.

The following sections describe Event Manager and the DECevent utility.

3.1.1 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 applications 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.

3.1.2 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 attempts to isolate failing device components and provides a notification mechanism that can warn of potential problems.

DECevent determines if a threshold has been crossed, according to the number and severity of events reported. Depending on the type of threshold crossed, DECevent analyzes the events and notifies users of the events (for example, through mail).

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.

3.2 Using System Accounting and Disk Quotas

It is recommended that you 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.

In addition, 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.

3.3 Continuously 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 in order to obtain accurate performance information.

The following tools allow you to continuously monitor performance:

Table 3–1: Tools for Continuous Performance Monitoring

Name	Description
Performance Manager	Simultaneously monitors multiple Tru64 UNIX systems, detects performance problems, and performs event notification. See Section 3.3.1 for more information.
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. See Section 3.3.2 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 `ftp://gatekeeper.dec.com/pub/DEC` for 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.
`tcpdump`	Continuously monitors the network traffic associated with a particular network service and allows you to identify the source of a packet. See `tcpdump`(8) for information.
`nfswatch`	Continuously monitors all incoming network traffic to a Network File System (NFS) server, and displays the number and percentage of packets received. See `nfswatch`(8) for information.
`xload`	Displays the system load average in a histogram that is periodically updated. See `xload`(1X) for information.

Table 3–1: 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 8.4.7.2 for information.
`volwatch`	Monitors LSM for failures in disks, volumes, and plexes, and sends mail if a failure occurs. See Section 8.4.7.4 for information.

The following sections describe the Performance Manager and Performance Visualizer products.

3.3.1 Using Performance Manager

Performance Manager (PM) for Tru64 UNIX allows you to simultaneously monitor multiple Tru64 UNIX systems, so you can detect and correct performance problems. PM can operate in the background, alerting you to performance problems. Monitoring only a local node does not require a PM license. However, a license is required to monitor multiple nodes and clusters.

Performance Manager (PM) is located on Volume 2 of the Associated Products CD-ROM in your distribution kit. To use PM, you must be running the `pmgrd` daemon. To start PM, invoke the `/usr/bin/pmgr` command.

PM gathers and displays Simple Network Protocol (SNMP and eSNMP) data for the systems you choose, and allows you to detect and correct performance problems from a central location. PM has a graphical user interface (GUI) that runs locally and displays data from the monitored systems. Use the GUI to choose the systems and data that you want to monitor.

You can customize and extend PM to create and save performance monitoring sessions. Graphs and charts can show hundreds of different system values, including CPU performance, memory usage, disk transfers, file-system capacity, network efficiency, database performance, and AdvFS and cluster-specific metrics. Data archives can be used for high-speed playback or long-term trend analysis.

PM provides comprehensive thresholding, rearming, and tolerance facilities for all displayed metrics. You can set a threshold on every key metric, and specify the PM reaction when a threshold is crossed. For example, you can configure PM to send mail, to execute a command, or to display a notification message.

PM also has performance analysis and system management scripts, as well as cluster-specific and AdvFS-specific scripts. Run these scripts separately to target specific problems, or run them simultaneously to check the overall system performance. The PM analyses include suggestions for eliminating problems. PM can monitor both individual cluster members and an entire cluster concurrently.

See `http://www.zso.dec.com/unix/pm/pmweb/index.html` for information about Performance Manager.

3.3.2 Using Performance Visualizer

Performance Visualizer is a valuable tool for developers of parallel applications. Because 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.

Performance Visualizer also helps you identify overloaded systems, underutilized resources, active users, and busy processes. You can monitor the following:

- CPU utilization by each CPU in a multiprocessing system
- Load average
- Use of paged memory
- Paging events, which indicate how much a system is paging
- Use of swap space
- Behavior of individual 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.

3.4 Gathering Performance Information

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 to 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.

To obtain a basic understanding of system performance, invoke the following commands while under a normal workload:

- `vmstat`

  The primary source of performance problems is a lack of memory, which can affect response time and application completion time. Use the `vmstat` command to monitor memory usage. If the command output shows a low free page count or excessive page outs, you may be overallocating memory. See Section 4.4.2 for information about solving low memory problems.

  In addition, a lack of CPU resources can result in long application completion times. The `vmstat` command enables you to monitor CPU usage. If the command output shows a high system or user time, see Section 4.4.5 for information about solving CPU resource problems.

  See Section 6.3.1 for information about using the the `vmstat` command to monitor memory and CPU usage.

- `iostat`, `volstat`, and `advfsstat`

  If your disk I/O load is not spread evenly among the available disks, bottlenecks may occur at the disks that are being excessively used. Use the `iostat`, `volstat` command (for Logical Storage Manager), and `advfsstat` (for Advanced File System) commands to determine if disk I/O is being evenly distributed. If the command output shows that some disks are being excessively used, see Section 4.4.6 for information about identifying and relieving disk bottlenecks.

  See Section 8.2 for information about monitoring the distribution of disk I/O.

- `swapon -s`

  Insufficient swap space for your workload can result in poor application performance and response time. To check swap space, use the `swapon -s` command. If the command output shows insufficient swap space, see Section 4.4.3 for information about increasing swap resources.

  See Section 6.3.3 for information about monitoring swap space.

There are many tools that you can use to query subsystems, profile the system kernel and applications, and collect CPU statistics. See the following tables for information:

Kernel profiling and debugging	Table 3–2
Memory resource monitoring	Table 6–2
CPU monitoring	Table 7–1
Disk I/O distribution monitoring	Table 8–1
Logical Storage Manager (LSM) monitoring	Table 8–7
Advanced File System (AdvFS) monitoring	Table 9–3
UNIX File System (UFS) monitoring	Table 9–7
Network File System (NFS) monitoring	Table 9–9
Network subsystem monitoring	Table 10–1
Application profiling and debugging	Table 11–1

3.5 Profiling and Debugging Kernels

Table 3–2 describes the tools that you can use to profile and debug the kernel. Detailed information about these profiling and debugging tools is located in the *Kernel Debugging* manual and in the tools' reference pages.

Table 3–2: Kernel Profiling and Debugging Tools

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`. See `prof`(1) for more information.
`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.

Table 3–2: Kernel Profiling and Debugging Tools (cont.)

Name	Use	Description
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.
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.

3.6 Accessing and Modifying Kernel Subsystems

The operating system includes various subsystems that are used to define or extend the kernel. Kernel variables control subsystem behavior or track subsystem statistics since boot time.

Kernel variables are assigned default values at boot time. For certain configurations and workloads, especially memory- or network-intensive systems, the default values of some attributes may not be appropriate, so you must modify these values to provide optimal performance.

Although you can use the dbx debugger to directly change variable values on a running kernel, Compaq recommends that you use kernel subsystem attributes to access the kernel variables.

Subsystem attributes are managed by the configuration manager server, cfgmgr. You can display and modify attributes by using the sysconfig and sysconfigdb commands and by using the Kernel Tuner, dxkerneltuner, which is provided by the Common Desktop Environment (CDE). In some cases, you can modify attributes while the system is running. However, these run-time modifications are lost when the system reboots.

The following sections describe how to perform these tasks:

- Display the subsystems configured in a system (Section 3.6.1)
- Display the current values of subsystem attributes (Section 3.6.2)
- Display the maximum and minimum values of subsystem attributes (Section 3.6.3)
- Modify current attribute values at run time(Section 3.6.4)
- Modify attribute values at boot time (Section 3.6.5)
- Permanently modify attribute values (Section 3.6.6)
- Use dbx to display and modify kernel variables (Section 3.6.7)

3.6.1 Displaying the Subsystems Configured in the Kernel

Each system includes different subsystems, depending on the configuration and the installed kernel options. For example, all systems include the mandatory subsystems, such as the generic, vm, and vfs subsystems. Other subsystems are optional, such as the Prestoserve subsystem presto.

Use one of the following methods to display the kernel subsystems currently configured in your operating system:

- sysconfig -s

  This command displays the subsystems currently configured in the operating system. See sysconfig(8) for more information.

- Kernel Tuner

  This GUI displays the subsystems currently configured in the operating system. To access the Kernel Tuner, click on the Application Manager icon in the CDE menu bar, select System_Admin, and then select MonitoringTuning. You can then click on Kernel Tuner. A pop-up menu containing a list of configured subsystems appears.

The following example shows how to use the sysconfig -s command to display the subsystems configured in the kernel:

```
# sysconfig -s
cm: loaded and configured
hs: loaded and configured
ksm: loaded and configured
generic: loaded and configured
io: loaded and configured
ipc: loaded and configured
proc: loaded and configured
sec: loaded and configured
socket: loaded and configured
rt: loaded and configured
bsd_tty: loaded and configured
xpr: loaded and configured
kdebug: loaded and configured
dli: loaded and configured
ffm_fs: loaded and configured
atm: loaded and configured
atmip: loaded and configured
lane: loaded and configured
atmifmp: loaded and configured
atmuni: loaded and configured
atmilmi3x: loaded and configured
uni3x: loaded and configured
bparm: loaded and configured
advfs: loaded and configured
net: loaded and configured
 .
 .
 .
```

3.6.2 Displaying Current Subsystem Attribute Values

Most kernel subsystems include one or more attributes. These attributes control or monitor some part of the subsystem and are assigned a value at boot time. For example, the `vm` subsystem includes the `vm_page_free_swap` attribute, which controls when swapping starts. The `socket` subsystem includes the `sobacklog_hiwat` attribute, which monitors the maximum number of pending socket requests.

Kernel subsystem attributes are documented in the reference pages. For example, `sys_attrs_advfs`(5) includes definitions for all the `advfs` subsystem attributes. See `sys_attrs`(5) for more information.

Use one of the following methods to display the current value of an attribute:

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

  This command displays the current values for the attributes of the specified subsystem or the current value for only the specified attribute. See `sysconfig`(8) for more information.

- Kernel Tuner

  This GUI displays the current values of all the subsystem attributes. To access the Kernel Tuner, click on the Application Manager icon in the CDE menu bar, select System_Admin, and then select MonitoringTuning. You can then click on Kernel Tuner. A pop-up menu containing a list

of subsystems appears. Select a subsystem to display the subsystem attributes and their current values.

The following example shows how to use the `sysconfig -q` command to display the current values of the `vfs` subsystem attributes:

```
# sysconfig -q vfs
vfs:
name_cache_size = 3141
name_cache_hash_size = 512
buffer_hash_size = 2048
special_vnode_alias_tbl_size = 64
bufcache = 3
bufpages = 1958
path_num_max = 64
sys_v_mode = 0
ucred_max = 256
nvnode = 1428
max_vnodes = 49147
min_free_vnodes = 1428
vnode_age = 120
namei_cache_valid_time = 1200
max_free_file_structures = 0
max_ufs_mounts = 1000
vnode_deallocation_enable = 1
pipe_maxbuf_size = 262144
pipe_databuf_size = 8192
pipe_max_bytes_all_pipes = 134217728
noadd_exec_access = 0
fifo_do_adaptive = 1
nlock_record = 10000
smoothsync_age = 30
revoke_tty_only = 1
strict_posix_osync = 0
```

Note

The current value of an attribute may not reflect a legal value, if you are not actually using a subsystem.

3.6.3 Displaying Minimum and Maximum Attribute Values

Each subsystem attribute has a minimum and maximum value. If you modify an attribute, the value must be between these values. However, the minimum and maximum values should be used with caution. Instead, use the tuning guidelines described in this manual to determine an appropriate attribute value for your configuration.

Use one of the following methods to display the minimum and maximum allowable values for an attribute:

- `sysconfig -Q` *subsystem* [*attribute*]

  Displays the minimum and maximum values for all the attributes of the specified subsystem or for only the specified attribute. The command output includes information in the `type` field about the value expected

(for example, integer, unsigned integer, long integer, or string). The output also specifies, in the op field, the operations that you can perform on the attribute:

- C—The attribute can be modified only when the subsystem is initially loaded; that is, the attribute supports only 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 is 0 and the maximum value is 50. The output also shows that you cannot modify the current (run-time) value.

See sysconfig(8) for more information.

- Kernel Tuner

  This GUI displays the minimum and maximum values for the subsystem attributes. To access the Kernel Tuner, click on the Application Manager icon in the CDE menu bar, select System_Admin, and then select MonitoringTuning. You can then click on Kernel Tuner. A pop-up menu containing a list of subsystems appears. Select a subsystem to display a list of the subsystem's attributes and their minimum and maximum values.

3.6.4 Modifying Attribute Values at Run Time

Modifying an attribute's current value at run time allows the change to occur immediately, without rebooting the system. Not all attributes support run-time modifications.

Modifications to run-time values are lost when you reboot the system and the attribute values return to their permanent values. To make a permanent change to an attribute value, see Section 3.6.6.

To determine if an attribute can be tuned at run time, use one of the following methods:

- sysconfig -Q *subsystem* [*attribute*]

  The command output indicates that an attribute can be tuned at run time if the op field includes an R. The following command shows that the max_vnodes attribute can be tuned at run time:

```
# sysconfig -Q vfs max-vnodes
vfs:
max_vnodes -    type=INT op=CRQ min_val=0 max_val=1717986918
```

See sysconfig(8) for more information.

- Kernel Tuner

  To access the Kernel Tuner, click on the Application Manager icon in the CDE menu bar, select System_Admin, and then select MonitoringTuning. You can then click on Kernel Tuner. A pop-up menu containing a list of subsystems appears. Select a subsystem to display the subsystem's attributes. The attribute can be tuned at run time if the attribute's Current Value field allows you to enter a value.

To modify an attribute's value at run time, use one of the following methods:

- sysconfig -r *subsystem attribute=value*

  This command modifies the run-time value of the specified subsystem attribute. The *value* argument specifies the new attribute value that you want the system to use. The following example changes the current value of the socket subsystem attribute somaxconn to 65536.

```
# sysconfig -r socket somaxconn=65535
somaxconn: reconfigured
```

- Kernel Tuner

  This GUI allows you to enter a new current value for an attribute. To access the Kernel Tuner, click on the Application Manager icon in the CDE menu bar, select System_Admin, and then select MonitoringTuning. You can then click on Kernel Tuner. A pop-up menu containing a list of subsystems appears. Select a subsystem and enter the new attribute value in the attribute's Current Value field.

Note

Do not specify erroneous values for subsystem attributes, because system behavior may be unpredictable. If you want to modify an attribute, use only the recommended values described in this manual.

To return to the original attribute value, either modify attribute's current value or reboot the system.

3.6.5 Modifying Attribute Values at Boot Time

You can set the value of a subsystem attribute at the boot prompt. This will modify the value of the attribute only for the next system boot.

To do this, enter the `b -fl i` command at the boot prompt. You will be prompted for the kernel name, attribute name, and attribute value. For example, enter the following to set the value of the `somaxconn` attribute to `65535`:

```
Enter kernel_name [option_1 ... option_n]: vmunix somaxconn=65535
```

3.6.6 Permanently Modifying Attribute Values

To permanently change the value of an attribute, you must include the new value in the `/etc/sysconfigtab` file, using the required format. Do not edit the file manually.

Note

Before you permanently modify a subsystem attribute, it is recommended that you maintain a record of the original value, in case you need to return to this value.

Use one of the following methods to permanently modify the value of an attribute:

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

  Use this command to specify the stanza-formatted file that contains the subsystem, the attribute, and the new permanent attribute value. The *subsystem* argument specifies the subsystem whose attribute you want to modify. See `stanza`(4) and `sysconfigdb`(8) for more information.

  The following is an example of a stanza-formatted file that changes the permanent values of the `socket` subsystem attributes `somaxconn` and `somaxconn` to `65535`:

  ```
  socket:
   somaxconn = 65535
   sominconn = 65535
  ```

  See `stanza`(4) for information about stanza-formatted files.

  To use the new permanent value, reboot the system or, if the attribute can be tuned at run time, use the `sysconfig -r` command to change the current value (see Section 3.6.4).

- Kernel Tuner

  This GUI allows you to change the permanent value of an attribute. To access the Kernel Tuner, click on the Application Manager icon in the CDE menu bar, select System_Admin, and then select MonitoringTuning. You can then click on Kernel Tuner. A pop-up menu containing a list of subsystems appears. Select the subsystem for the attribute that you want to modify. Enter the new permanent value in the attribute's `Boot Time Value` field.

To use the new attribute value, reboot the system or, if the attribute can be tuned at run time, enter the new value in the Current Value field (see Section 3.6.4).

Note

Do not specify erroneous values for subsystem attributes, because system behavior may be unpredictable. If you want to modify an attribute, use only the recommended values described in this manual.

3.6.7 Displaying and Modifying Kernel Variables by Using the dbx Debugger

Use the dbx debugger to examine the values of kernel variables and data structures, and to modify the current (run-time) values of kernel variables.

Note

In some cases, you must specify the processor number with the dbx print command. For example, to examine the nchstats data structure on a single-processor system, use the dbx print processor_ptr[0].nchstats command.

The following example of the dbx print command displays the current (run-time) value of the maxusers kernel variable:

```
# /usr/ucb/dbx -k /vmunix /dev/mem
(dbx) print maxusers
512
(dbx)
```

Use the dbx patch command to modify the current (run-time) values of kernel variables. The values you assign by using the dbx patch command are lost when you rebuild the kernel.

Notes

If possible, use the sysconfig command or the Kernel Tuner to modify subsystem attributes instead of using dbx to modify kernel variables. Do not specify erroneous values for kernel variables, because system behavior may be unpredictable. If you want to modify a variable, use only the recommended values described in this manual.

The following example of the `dbx patch` command changes the current value of the `cluster_consec_init` variable to `8`:

```
# /usr/ucb/dbx –k /vmunix /dev/mem
(dbx) patch cluster_consec_init = 8
8
(dbx)
```

To ensure that the system is utilizing a new kernel variable value, reboot the system. See the *Programmer's Guide* for detailed information about the `dbx` debugger.

You can also use the `dbx assign` command to modify run-time kernel variable values. However, the modifications are lost when you reboot the system.

4

Improving System Performance

You may be able to improve Tru64 UNIX performance by tuning the operating system or performing other tasks. You may need to tune the system under the following circumstances:

- You are running a large or specialized configuration that requires you to modify the default values of some subsystem attributes.
- You want to optimize performance in a generally well-functioning system.
- You want to solve a specific performance problem.

To help you improve system performance, this chapter describes the following:

- Steps for configuring and tuning high-performance and high-availability systems (see Section 4.1)
- Applying configuration-specific tuning guidelines (Section 4.2)
- Running `sys_check` and applying its configuration and tuning guidelines (Section 4.3)
- Identifying and solving some common performance problems (Section 4.4)
- Using the advanced tuning guidelines described in this manual (Section 4.5)

4.1 Steps for Configuring and Tuning Systems

Before you configure and tune a system, you must become familiar with the terminology and concepts relating to performance and availability. See Chapter 1 for information.

In addition, you must understand how your applications utilize 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 2.1 for information about identifying a resource model for your configuration.

To help you configure and tune a system that will meet your performance and availability needs, follow these steps:

1. Ensure that your hardware and software configuration is appropriate for your workload resource model and your performance and availability goals. See Chapter 2.
2. Make sure that you have adhered to the configuration guidelines for:
   - Memory and swap space (Section 2.3.2)
   - Disks, LSM, and hardware RAID (Chapter 8)
   - AdvFS, UFS, and NFS file systems (Chapter 9)
3. Perform the following initial tuning tasks:
   a. If you have a large-memory system, Internet server, or NFS server, follow the tuning guidelines that are described in Section 4.2.
   b. Apply any tuning recommendations described in your application documentation.
   c. Make sure that you have sufficient system resources for large applications or for large-memory systems. See Chapter 5 for information about resource tuning.
   d. Run `sys_check` and consider following its configuration and tuning recommendations (see Section 4.3).
4. Monitor the system and evaluate its performance, identifying any areas in which performance can be improved. Section 3.4 describes the tools that you can use to monitor performance.
5. If performance is deficient, see Section 4.4 for information about solving common performance problems, and see Section 4.5 for information about using the advanced tuning guidelines.

System tuning usually involves modifying kernel subsystem attributes. See Section 3.6 for information.

4.2 Tuning Special Configurations

Large configurations or configurations that run memory-intensive or network-intensive applications may require special tuning. The following sections provide information about tuning these special configurations:

- Internet servers (Section 4.2.1)
- Large-memory servers (Section 4.2.2)
- NFS servers (Section 4.2.3)

In addition, your application product documentation may include specific configuration and tuning guidelines that you should follow.

4.2.1 Tuning Internet Servers

Internet servers (including Web, proxy, firewall, and gateway servers) run network-intensive applications that usually require significant system resources. If you have an Internet server, it is recommended that you modify the default values of some kernel attributes.

Follow the guidelines in Table 4–1 to help you tune an Internet server.

Table 4–1: Internet Server Tuning Guidelines

Guideline	Reference
Increase the system resources available to processes.	Section 5.1
Increase the available address space.	Section 5.3
Ensure that the Unified Buffer Cache (UBC) has sufficient memory.	Section 9.2.4
Increase the size of the hash table that the kernel uses to look up TCP control blocks.	Section 10.2.1
Increase the number of TCP hash tables.	Section 10.2.2
Increase the limits for partial TCP connections on the socket listen queue.	Section 10.2.3
For proxy servers only, increase the maximum number of concurrent nonreserved, dynamically allocated ports.	Section 10.2.4
Disable use of a path maximum transmission unit (PMTU).	Section 10.2.6
Increase the number of IP input queues.	Section 10.2.7
For proxy servers only, enable `mbuf` cluster compression.	Section 10.2.8

4.2.2 Tuning Large-Memory Systems

Large memory systems often run memory-intensive applications, such as database programs, that usually require significant system resources. If you have a large memory system, it is recommended that you modify the default values of some kernel attributes.

Follow the guidelines in Table 4–2 to help you tune a large-memory system.

Table 4–2: Large-Memory System Tuning Guidelines

Guideline	Reference
Increase the system resources available to processes.	Section 5.1
Increase the size of a System V message and queue.	Section 5.4.1
Increase the maximum size of a single System V shared memory region.	Section 5.4.4
Increase the minimum size of a System V shared memory segment.	Section 5.4.6
Increase the available address space.	Section 5.3
Reduce the size of the AdvFS buffer cache.	Section 6.4.4
Increase the number of AdvFS buffer hash chains, if you are using AdvFS.	Section 9.3.6.2
Increase the memory reserved for AdvFS access structures, if you are using AdvFS.	Section 9.3.6.3
Increase the size of the metadata buffer cache to more than 3 percent of main memory, if you are using UFS.	Section 9.4.3.1
Increase the size of the metadata hash chain table, if you are using UFS.	Section 9.4.3.2

4.2.3 Tuning NFS Servers

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 9 for information on file system tuning.

In addition, if you are running NFS over 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 UDP, network subsystem tuning is not needed.

Follow the guidelines in Table 4–3 to help you tune a system that is only serving NFS.

Table 4–3: 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 5.1
Increase the size of the namei cache.	Section 9.2.1

Table 4–3: NFS Server Tuning Guidelines (cont.)

Guideline	Reference
Increase the memory reserved for AdvFS access structures, if you are using AdvFS.	Section 9.3.6.3
Increase the size of the metadata buffer cache, if you are using UFS.	Section 9.4.3.1

4.3 Checking the Configuration by Using the sys_check Utility

After you apply any configuration-specific tuning guidelines, as described in Section 4.2, run the `sys_check` utility to check your system configuration.

The `sys_check` utility creates an HTML file that describes the system configuration, and can be used to diagnose problems. The utility checks kernel attribute settings and memory and CPU resources, provides performance data and lock statistics for SMP systems and for kernel profiles, and outputs any warnings and tuning guidelines.

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 following:

- 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.
- The `-escalate` option creates escalation files required for reporting problems to Compaq.

See `sys_check`(8) for more information.

4.4 Solving Common Performance Problems

The following sections provide examples of some common performance problems and solutions:

- Slow application performance (Section 4.4.1)
- Insufficient memory or excessive paging (Section 4.4.2)
- Insufficient swap space (Section 4.4.3)
- Swapped out processes (Section 4.4.4)
- Insufficient CPU cycles (Section 4.4.5)
- Disk bottleneck (Section 4.4.6)
- Poor disk I/O performance (Section 4.4.7)
- Poor AdvFS performance (Section 4.4.8)
- Poor UFS performance (Section 4.4.9)
- Poor NFS performance (Section 4.4.10)
- Poor network performance (Section 4.4.11)

Each section describes how to detect the problem, the possible causes of the problem, and how to eliminate or diminish the problem.

4.4.1 Application Completes Slowly

Use the following table to detect a slow application completion time and to diagnose the performance problem:

How to detect	Check application log files. Use the `ps` command to display information about application processing times and whether an application is swapped out. See Section 6.3.2. Use process accounting commands to obtain information about process completion times. See `accton`(8).
Cause	Application is inefficient.
Solution	Rewrite the application so that it runs more efficiently. See Chapter 7. Use profiling and debugging commands to analyze applications and identify inefficient areas of code. See Section 11.1.
Cause	Application is not optimized.
Solution	Optimize the application. See Chapter 7.
Cause	Application is being swapped out.
Solution	Delay swapping processes. See Section 6.5.3. Increase the memory available to processes. See Section 6.4. Reduce an application's use of memory. See Section 11.2.6.
Cause	Application requires more memory resources.
Solution	Increase the memory available to processes. See Section 6.4. Reduce an application's use of memory. See Section 11.2.6.
Cause	Insufficient swap space.
Solution	Increase the swap space and distribute it across multiple disks. See Section 4.4.3.
Cause	Application requires more CPU resources.
Solution	Provide more CPU resources to processes. See Section 4.4.5.
Cause	Disk I/O bottleneck.
Solution	Distribute disk I/O efficiently. See Section 4.4.6.

4.4.2 Insufficient Memory or Excessive Paging

A high rate of paging or a low free page count may indicate that you have inadequate memory for the workload. Avoid paging if you have a large memory system. Use the following table to detect insufficient memory and to diagnose the performance problem:

How to detect	Use the `vmstat` command to display information about paging and memory consumption. See Section 6.3.1 for more information.
Cause	Insufficient memory resources available to processes.
Solution	Reduce an application's use of memory. See Section 11.2.6. Increase the memory resources that are available to processes. See Section 6.4. Add physical memory.

4.4.3 Insufficient Swap Space

If you consume all the available swap space, the system will display messages on the console indicating the problem. Use the following table to detect if you have insufficient swap space and to diagnose the performance problem:

How to detect	Invoke the `swapon -s` while you are running a normal workload. See Section 6.3.3.
Cause	Insufficient swap space for your configuration.
Solution	Configure enough swap space for your configuration and workload. See Section 2.3.2.3.
Cause	Swap space not distributed.
Solution	Distribute the swap load across multiple swap devices to improve performance. See Section 6.2.
Cause	Applications are utilizing excessive memory resources.
Solution	Increase the memory available to processes. See Section 6.4. Reduce an application's use of memory. See Section 11.2.6.

4.4.4 Processes Swapped Out

Swapped out (suspended) processes will decrease system response time and application completion time. Avoid swapping if you have a large memory system or large applications. Use the following table to detect if processes are being swapped out and to diagnose the performance problem:

How to detect	Use the `ps` command to determine if your system is swapping processes. See Section 6.3.2.
Cause	Insufficient memory resources.
Solution	Increase the memory available to processes. See Section 6.4. Reduce an application's use of memory. See Section 11.2.6.
Cause	Swapping occurs too early during page reclamation.
Solution	Decrease the rate of swapping. See Section 6.5.3.

4.4.5 Insufficient CPU Cycles

Although a low CPU idle time can indicate that the CPU is being fully utilized, performance can suffer if the system cannot provide a sufficient number of CPU cycles to processes. Use the following table to detect insufficient CPU cycles and to diagnose the performance problem:

How to detect	Use the `vmstat` command to display information about CPU system, user, and idle times. See Section 6.3.1 for more information. Use the `kdbx cpustat` extension to check CPU usage. See Section 7.1.4).
Cause	Excessive CPU demand from applications.
Solution	Optimize applications. See Section 11.2.4. Use hardware RAID to relieve the CPU of disk I/O overhead. See Section 8.5. Add processors

4.4.6 Disk Bottleneck

Excessive I/O to only one or a few disks may cause a bottleneck at the overutilized disks. Use the following table to detect an uneven distribution of disk I/O and to diagnose the performance problem:

How to detect	Use the `iostat` command to display which disks are being used the most. See Section 8.2. Use the `swapon -s` command to display the utilization of swap disks. See Section 6.3.3. Use the `volstat` command to display information about the LSM I/O workload. See Section 8.4.7.2 for more information. Use the `advfsstat` to display AdvFS disk usage information. See Section 9.3.5.1.
Cause	Disk I/O not evenly distributed.
Solution	Use disk striping. See Section 2.5.2. Distribute disk, swap, and file system I/O across different disks and, optimally, multiple buses. See Section 8.1.

4.4.7 Poor Disk I/O Performance

Because disk I/O operations are much slower than memory operations, the disk I/O subsystem is often the source of performance problems. Use the following table to detect poor disk I/O performance and to diagnose the performance problem:

How to detect	Monitor the memory allocated to the UBC by using the `dbx ufs_getapage_stats` and `vm_tune` data structures. See Section 6.3.4. Use the `iostat` command to determine if a you have a bottleneck at a disk. See Section 8.2 for more information. Check for disk fragmentation. See Section 9.3.7.1 and Section 9.4.3.7. Check the hit rate of the namei cache with the `dbx nchstats` data structure. See Section 9.1.2. Use the `advfsstat` command to monitor the performance of AdvFS domains and filesets. See Section 9.3.5.1. Check UFS clustering with the `dbx ufs_clusterstats` data structure. See Section 6.3.4. Check the hit rate of the metadata buffer cache by using the `dbx bio_stats` data structure. See Section 9.4.2.3.
Cause	Disk I/O is not efficiently distributed.
Solution	Use disk striping. See Section 2.5.2. Distribute disk, swap, and file system I/O across different disks and, optimally, multiple buses. See Section 8.1.
Cause	File systems are fragmented.
Solution	Defragment file systems. See Section 9.3.7.1 and Section 9.4.3.7.
Cause	Maximum open file limit is too small.
Solution	Increase the maximum number of open files. See Section 5.5.1.
Cause	The namei cache is too small.
Solution	Increase the size of the namei cache. See Section 9.2.1.

4.4.8 Poor AdvFS Performance

Use the following table to detect poor AdvFS performance and to diagnose the performance problem:

How to detect	Use the `advfsstat` command to monitor the performance of AdvFS domains and filesets. See Section 9.3.5.1. Check for disk fragmentation by using the AdvFS `defragment` command with the `-v` and `-n` options. See Section 9.3.7.1.
Cause	Single-volume domains are being used.
Solution	Use multiple-volume file domains. See Section 9.3.4.1.
Cause	File system is fragmented.
Solution	Defragment the file system. See Section 9.3.7.1.
Cause	There are too few AdvFS buffer cache hits.
Solution	Allocate sufficient memory to the AdvFS buffer cache. See Section 9.3.6.1. Increase the number of AdvFS buffer hash chains (Section 9.3.6.2. Increase the dirty data caching threshold. See Section 9.3.6.4. Modify the AdvFS device queue limit. See Section 9.3.6.6.
Cause	The `advfsd` daemon is running unnecessarily.
Solution	Stop the daemon. See Section 7.2.5.

4.4.9 Poor UFS Performance

Use the following table to detect poor UFS performance and to diagnose the performance problem:

How to detect	Monitor the memory allocated to the UBC by using the `dbx ufs_getapage_stats`. See Section 6.3.4. Check the hit rate of the namei cache with the `dbx nchstats` data structure. See Section 9.1.2. Use the `dumpfs` command to display UFS information. See Section 9.4.2.1. Check how effectively the system is clustering and check fragmentation by using the `dbx print` command to examine the `ufs_clusterstats`, `ufs_clusterstats_read`, and `ufs_clusterstats_write` data structures. See Section 9.4.2.2. Check the hit rate of the metadata buffer cache by using the `dbx bio_stats` data structure. See Section 9.4.2.3.
Cause	The UBC is too small.
Solution	Increase the amount of memory allocated to the UBC. See Section 9.2.4.
Cause	The metadata buffer cache is too small.
Solution	Increase the size of metadata buffer cache. See Section 9.4.3.1.
Cause	The file system fragment size is incorrect.
Solution	Make the file system fragment size equal to the block size. See Section 9.4.1.1.
Cause	File system is fragmented.
Solution	Defragment the file system. Section 9.4.3.7.

4.4.10 Poor NFS Performance

Use the following table to detect poor NFS performance and to diagnose the performance problem:

How to detect	Use the `dbx print nfs_sv_active_hist` command to display a histogram of the active NFS server threads. See Section 3.6.7. Use the `dbx print nchstats` command to determine the namei cache hit rate. See Section 9.1.2. Use the `dbx print bio_stats` command to determine the metadata buffer cache hit rate. See Section 9.4.2.3. Use the `nfsstat` command to display the number of NFS requests and other information. See Section 9.5.1.1. Use the `ps axlmp 0 | grep nfs` command to display the number of idle threads. See Section 9.5.2.3.
Cause	NFS server threads busy.
Solution	Reconfigure the server to run more threads. See Section 9.5.2.2.
Cause	Memory resources are not focused on file system caching.
Solution	Increase the amount of memory allocated to the UBC. See Section 9.2.4. If you are using AdvFS, increase the memory allocated for AdvFS buffer caching. See Section 9.3.6.1. If you are using AdvFS, increase the memory reserved for AdvFS access structures. See Section 9.3.6.3 for information.
Cause	System resource allocation is not adequate.
Solution	Set the value of the `maxusers` attribute to the number of server NFS operations that are expected to occur each second. See Section 5.1 for information.
Cause	UFS metadata buffer cache hit rate is low.
Solution	Increase the size of the metadata buffer cache. See Section 9.4.3.1. Increase the size of the namei cache. See Section 9.2.1.
Cause	CPU idle time is low.
Solution	Use UFS, instead of AdvFS. See Section 9.4.

4.4.11 Poor Network Performance

Use the following table to detect poor network performance and to diagnose the performance problem:

How to detect	Use the `netstat` command to display information about network collisions and dropped network connections. See Section 10.1.1. Check the socket listen queue statistics to check the number of pending requests and the number of times the system dropped a received SYN packet. See Section 10.1.2.
Cause	The TCP hash table is too small.
Solution	Increase the size of the hash table that the kernel uses to look up TCP control blocks. See Section 10.2.1.
Cause	The limit for the socket listen queue is too low.
Solution	Increase the limit for partial TCP connections on the socket listen queue. See Section 10.2.3.
Cause	There are too few outgoing network ports.
Solution	Increase the maximum number of concurrent nonreserved, dynamically allocated ports. See Section 10.2.4.
Cause	Network connections are becoming inactive too quickly.
Solution	Enable TCP keepalive functionality. See Section 10.2.9.

4.5 Using the Advanced Tuning Guidelines

If system performance is still deficient after applying the initial tuning recommendations (Section 4.1) and considering the solutions to common performance problems (Section 4.4), you may be able to improve performance by using the advanced tuning guidelines. Advanced tuning requires an in-depth knowledge of Tru64 UNIX and the applications running on the system, and should be performed by an experienced system administrator.

Before using the advanced tuning guidelines, you must:

- Understand your workload resource model, because not all tuning guidelines are appropriate for all configurations (see Section 2.1)
- Gather performance information to identify an area in which to focus your efforts (see Section 3.4 for information on using commands to obtain a basic understanding of system performance)

Use the advanced tuning guidelines shown in Table 4–4 to help you tune your system. Before implementing any tuning guideline, you must ensure that it is appropriate for your configuration and workload and also consider its benefits and tradeoffs.

Table 4–4: Advanced Tuning Guidelines

If your workload consists of:	You can improve performance by:
Applications requiring extensive system resources	Increasing resource limits (Chapter 5)
Memory-intensive applications	Increasing the memory available to processes (Section 6.4) Modifying paging and swapping operations (Section 6.5) Reserving shared memory (Section 6.6)
CPU-intensive applications	Freeing CPU resources (Section 7.2)
Disk I/O-intensive applications	Distributing the disk I/O load (Section 8.1)
File system-intensive applications	Modifying AdvFS, UFS, or NFS operation (Chapter 9)
Network-intensive applications	Modifying network operation (Section 10.2)
Nonoptimized or poorly written applications applications	Optimizing or rewriting the applications (Chapter 11)

5

Tuning 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 how to increase the following system-wide limits:

- Process limits (Section 5.1)
- Program size limits (Section 5.2)
- Address space limits (Section 5.3)
- Interprocess communication (IPC) limits (Section 5.4)
- Open file limits (Section 5.5)

Instead of modifying system-wide 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 information.

5.1 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 5.1.1)
- Maximum number of processes (Section 5.1.2)
- Maximum number of threads (Section 5.1.3)

5.1.1 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.

> 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_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 5–1 shows the default value of the `maxusers` attribute for systems with various amounts of memory.

Table 5–1: 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
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. It is recommended 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 5.5.1).

You must not decrease the default value of the maxusers attribute.

See Section 3.6 for information about modifying kernel subsystem attributes.

5.1.2 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 (Section 5.1.1). 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 (zero) for the max_proc_per_user attribute, there is no limit on processes.

See Section 3.6 for information about modifying kernel subsystem attributes.

5.1.3 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 (Section 5.1.1). 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 sufficient 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 (zero) will remove the limit on threads.

If you specify a value of 0 (zero) for the max_threads_per_user attribute, there is no limit on threads.

See Section 3.6 for information about modifying kernel subsystem attributes.

5.2 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 perform the following tasks:

- Increase the maximum size of a user process stack (Section 5.2.1)

- Increase the maximum size of a user process data segment (Section 5.2.2)

5.2.1 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 8388608 bytes. The default value of the `max_per_proc_stack_size` attribute is 33554432 bytes. Choose values that are significantly less than the address space limit. See Section 5.3 for information.

See Section 3.6 for information about modifying kernel subsystem attributes.

5.2.2 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 1342177281 bites. The default value of the max_per_proc_data_size is 1 GB (1073741824 bytes). Choose values that are significantly less than the address space limit. See Section 5.3 for information.

If you have an Internet server, increase the value of the max_per_proc_data_size attribute to 10 GB (10737418240 bytes).

See Section 3.6 for information about modifying kernel subsystem attributes.

5.3 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, 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 (4294967296 bytes).

If you have an Internet server, increase the value of the `max_per_proc_address_space` attribute to 10 GB (10737418240 bytes).

See Section 3.6 for information about modifying kernel attributes.

5.4 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 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 5–2 describes the guidelines for increasing IPC limits and lists the performance benefits as well as tradeoffs.

Table 5–2: IPC Limits Tuning Guidelines

Guideline	Performance Benefit	Tradeoff
Increase the maximum size of a System V message (Section 5.4.1)	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 5.4.2)	May improve the performance of applications that can benefit from a large System V message queue	Consumes a small amount memory

Table 5–2: IPC Limits Tuning Guidelines (cont.)

Guideline	Performance Benefit	Tradeoff
Increase the maximum number of outstanding messages on a System V queue (Section 5.4.3)	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 5.4.4)	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 5.4.5)	May improve the performance of applications that attach many shared memory regions	May consume memory
Modify the shared page table limit (Section 5.4.6)	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.

5.4.1 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 Values

The default value of the `msg_max` attribute is 8192 bytes (1 page).

See Section 3.6 for information about modifying kernel subsystem attributes.

5.4.2 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 Values

The default value of the msg_mnb attribute is 16384 bytes.

See Section 3.6 for information about modifying kernel subsystem attributes.

5.4.3 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 Section 3.6 for information about modifying kernel subsystem attributes.

5.4.4 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 Values

The default value of the shm_max attribute is 4194304 bytes (512 pages).

See Section 3.6 for information about modifying kernel subsystem attributes.

5.4.5 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 Values

The default value of the `shm_seg` is 32.

See Section 3.6 for information about modifying kernel subsystem attributes.

5.4.6 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 Values

The default value of the ssm_threshold attribute is 8 MB (8388608 bytes).

Setting the ssm_threshold attribute to 0 (zero) will disable the use of segmented shared memory.

See Section 3.6 for information about modifying kernel subsystem attributes.

5.5 Tuning the Open File Limits

The following sections describe how to perform the following tasks:

- Increase the maximum number of open files (Section 5.5.1).
- Increase the maximum number of open file descriptors (Section 5.5.2).

5.5.1 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 handled dynamically 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 5.1 for 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 Values

The default value of the `max_vnodes` attribute is 5 percent of memory.

See Section 3.6 for information about modifying kernel subsystem attributes.

5.5.2 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 system-wide 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 4096, which is the maximum system-wide 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 system-wide 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 4096 (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 4096 for `FD_SETSIZE` in the `<sys/select.h>` header file by specifying the maximum value of 65536. 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 set explicitly 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. It is recommended 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 5.5.1 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 4096 open file descriptors. However, if the process is using more than 4096 file descriptors, the `setsysinfo` system call will return an `EINVAL` error. In addition, if a calling process's hard or soft limit exceeds 4096, the limit is set to 4 KB after the call is successful. This setting is inherited by any child process.

6

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. This can be done by increasing memory resources.

This chapter describes how to perform the following tasks:

- Understand how the operating system allocates virtual memory to processes and to file system caches, and how memory is reclaimed (Section 6.1)
- Configure swap space for high performance (Section 6.2)
- Obtain information about memory usage (Section 6.3)
- Provide more memory resources to processes (Section 6.4)
- Modify paging and swapping operation (Section 6.5)
- Reserve physical memory for shared memory (Section 6.6)

6.1 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 6.1.2.1).

- 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 6.1.1)
- Allocates memory to file system buffer caches (Section 6.1.2)
- Allocates memory to processes (Section 6.1.3)
- Reclaims pages (Section 6.1.4)

6.1.1 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 UBC

To determine which active pages should be reclaimed first, the page-stealer daemon identifies the oldest pages on the active list. The oldest pages that are being used by processes are designated **Inactive** pages. The oldest pages that are being used by the UBC are designates **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.

6.1.2 File System Buffer Cache Memory Allocation

The operating system uses three 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 6.1.2.1)
- Unified Buffer Cache (Section 6.1.2.2)
- AdvFS buffer cache (Section 6.1.2.3)

6.1.2.1 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 attribute `bufcache`. See Section 6.4.6 and Section 9.4.3.1 for tuning information.

6.1.2.2 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.

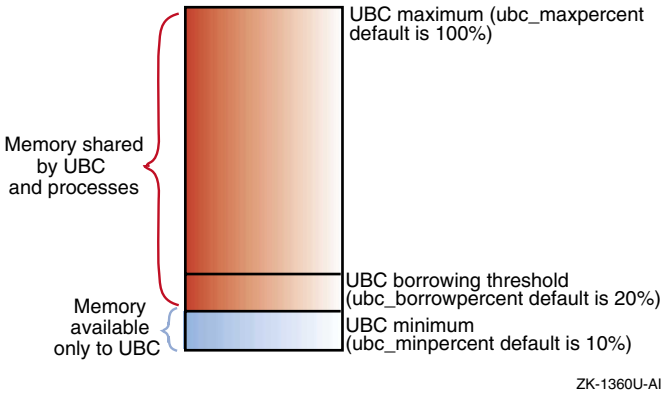
The UBC functions as a layer between the operating system and disk by storing recently accessed UFS file system data for reads and writes from conventional file activity and holding page faults from mapped file sections.

File system performance is improved if the cached data is later reused and a disk I/O operation is avoided.

In addition, AdvFS wires UBC pages for its metadata and file data, although the AdvFS buffer cache actively manages this data. See Section 6.1.2.3 for information about the AdvFS buffer cache.

Figure 6–1 shows how the virtual memory subsystem allocates physical memory to the UBC and for processes.

Figure 6–1: UBC Memory Allocation



The amount of memory that the UBC can utilize is determined by three `vm` subsystem attributes:

- `ubc_minpercent` attribute

  Specifies the minimum percentage of virtual memory that only the UBC can utilize. The remaining memory is shared with processes. The default is 10 percent.

- `ubc_maxpercent` attribute

  Specifies the maximum percentage of virtual memory that the UBC can utilize. The default is 100 percent.

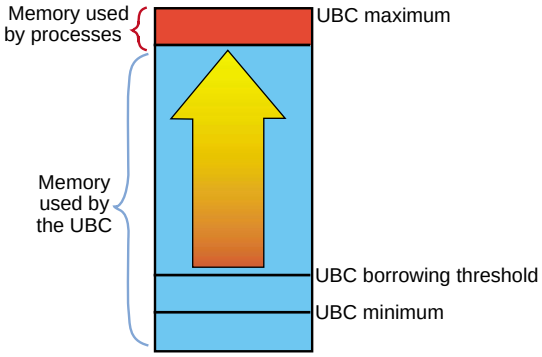
- `ubc_borrowpercent` attribute

  Specifies the UBC borrowing threshold. The default is 20 percent. 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, until the amount of memory allocated to the UBC decreases to the value of the `ubc_borrowpercent` attribute.

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

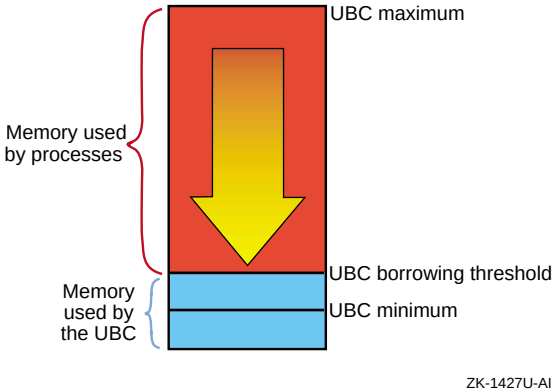
system activity is heavy and process demand is low, most of the pages will be allocated to the UBC, as shown in Figure 6–2.

Figure 6–2: 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 virtual memory subsystem to reclaim UBC borrowed pages, down to the value of the `ubc_borrowpercent` attribute, as shown in Figure 6–3.

Figure 6–3: Memory Allocation During Low File System Activity and High Paging Activity



The UBC uses a hashed list to quickly locate the physical pages that it is holding. A hash table contains file and offset information that is used to speed lookup operations.

The UBC also uses a buffer to facilitate the movement of data between memory and disk. The `vm` subsystem attribute `vm_ubcbuffers` specifies the

maximum file system device I/O queue depth for writes (that is, the number of UBC I/O requests that can be outstanding).

6.1.2.3 AdvFS Buffer Cache Memory Allocation

The AdvFS buffer cache functions as a layer between the operating system and disk by storing recently accessed file data and metadata. Performance is improved if the cached data is later reused and a disk operation is avoided.

AdvFS wires UBC pages for both file data and metadata, although AdvFS actively manages the cache by using its own management routines and data structures. For example, AdvFS performs its own hashing of cached data, and uses its own cache lookup routines and modified data flushing routines. AdvFS interacts with the UBC by using the UBC to hold the actual data content for metadata and file sysem user data. When an AdvFS I/O operation occurs, AdvFS searches the AdvFS buffer cache for the data before querying the UBC for the cache page.

You can tune the AdvFS buffer cache by modifying the values of some `advfs` attributes. Because AdvFS manages its own buffer cache, tuning the UBC will not have a great effect on AdvFS.

At boot time, the kernel determines the amount of physical memory that is available for AdvFS buffer cache headers, and allocates a buffer cache header for each possible page. Buffer headers are maintained in a global array and temporarily assigned a buffer handle that refers to an actual page. The number of AdvFS buffer headers depends on the number of 8-KB pages that can be obtained from the amount of memory specified by the `advfs` subsystem attribute `AdvfsCacheMaxPercent`. The default value is 7 percent of physical memory.

The AdvFS buffer cache is organized as a fixed-size hash chain table, which uses a file page offset, fileset handle, and domain handle to calculate the hash key that is used to look up a page. The size of the hash chain table depends on the number of buffer cache headers. However, you can override AdvFS table size calculation by changing the `AdvfsCacheHashSize` attribute.

When a page of data is requested, AdvFS searches the hash chain table for a match. If the entry is already in memory, AdvFS returns the buffer handle and a pointer to the page of data to the requester.

If no entry is found, AdvFS obtains a free buffer header and initializes it to represent the requested page. AdvFS performs a read operation to obtain the page from disk and attaches the buffer header to a UBC page. The UBC page is then wired into memory. AdvFS buffer cache pages remain wired until the buffer needs to be recycled, the file is deleted, or the fileset is unmounted.

See Section 6.4.4, Section 9.3.6.1, and Section 9.3.6.2 for information about tuning the AdvFS buffer cache.

6.1.3 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.

6.1.3.1 Process Virtual Address Space Allocation

The `fork` system call creates new processes. When you invoke a process, the `fork` system call:

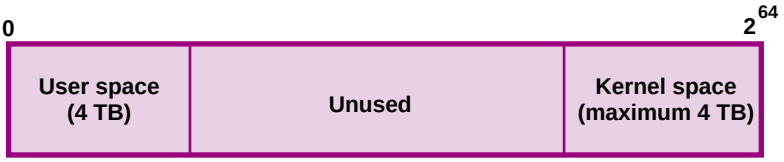
1. Creates a UNIX process body, which includes a set of data structures that the kernel uses to track the process and a set of resource limitations. See `fork`(2) for more information.
2. Establishes a contiguous block of **virtual address space** for the process. Virtual address space is the array of 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 6–4 shows the use of process virtual address space.

Figure 6–4: Virtual Address Space Usage

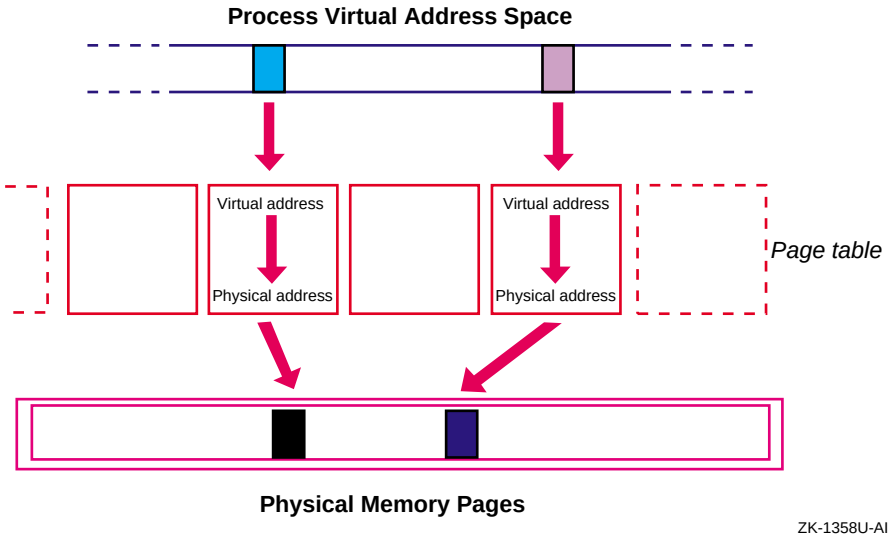


6.1.3.2 Virtual Address to Physical Address Translation

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 6–5 shows the translation of a virtual address into a physical address.

Figure 6–5: 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.

6.1.3.3 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 1–1).

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:

- 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:

  1. Allocates an available page of physical memory.
  2. Fills the page with zeros.
  3. Enters the virtual-to-physical address translation in the page table.

- 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.
- 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.

- 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 several 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

  If possible, the virtual memory subsystem maps 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.

6.1.4 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 6.1.4.1 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.

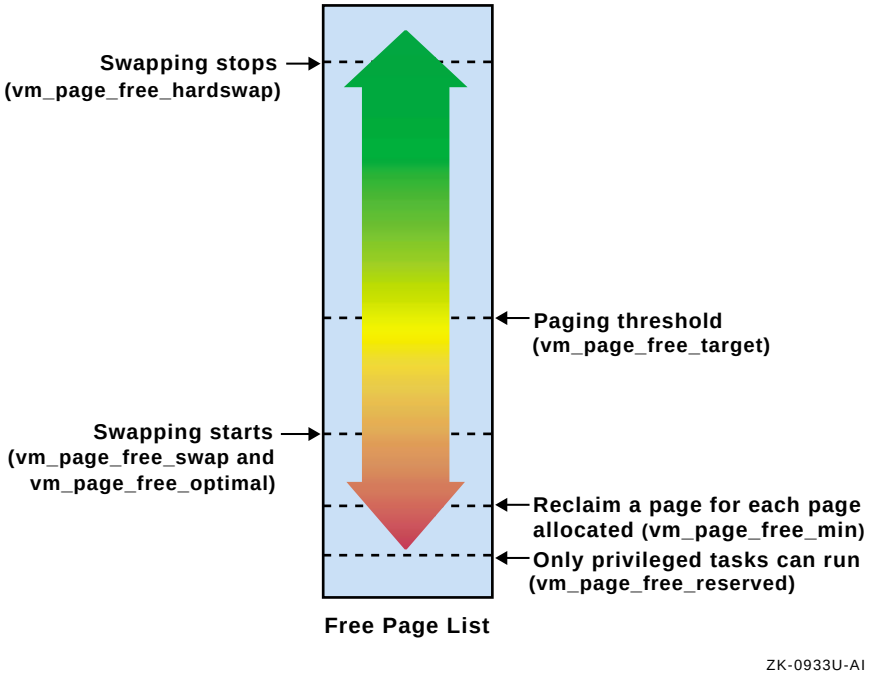
   b. Reclaims the oldest inactive and UBC LRU pages from the active page list, moves the contents of the modified pages to swap space or disk, and puts the clean pages on the free list.

   c. If needed, more aggressively reclaims pages from the active list.

   See Section 6.1.4.2 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 6.1.4.3 for information about swapping.

The point at which paging and swapping start and stop depends on the values of some vm subsystem attributes. Figure 6–6 shows some of the attributes that control paging and swapping.

Figure 6–6: Paging and Swapping Attributes



Detailed descriptions of the attributes are as follows:

- `vm_page_free_target`—Paging begins when the number of pages on the free list is less than this value. Paging stops when the number of pages is equal to or more than this value. The default value of the `vm_page_free_target` attribute is based on the amount of memory in the system. Use Table 6–1 to determine the default value for your system.

Table 6–1: 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 the threshold 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—Only privileged tasks can get memory when the number of pages on the free list is less than this value. The default value is 10 pages.
- vm_page_free_swap—Idle task swapping begins when the number of pages on the free list is less than this value for a period of time. The default value of the vm_page_free_swap attribute is based on the values of the vm_page_free_target and vm_page_free_min attributes by using this formula:

  ```
  vm_page_free_min + ((vm_page_free_target - vm_page_free_min) / 2)
  ```

- vm_page_free_optimal—Hard swapping begins when the number of pages on the free list is less than this value for five seconds. 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 of the vm_page_free_optimal attribute is based on the values of the vm_page_free_target and vm_page_free_min attributes by using this formula:

  ```
  vm_page_free_min + ((vm_page_free_target - vm_page_free_min) / 2)
  ```

- vm_page_free_hardswap—Swapping stops when the number of pages on the free list is equal to or more than this value. The default value is the value of the vm_page_free_target attribute multiplied by 16.

See Section 6.5 for information about modifying paging and swapping attributes.

The following sections describe the page reclamation procedure in detail.

6.1.4.1 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 6.1.1 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.

6.1.4.2 Reclaiming Memory by Paging

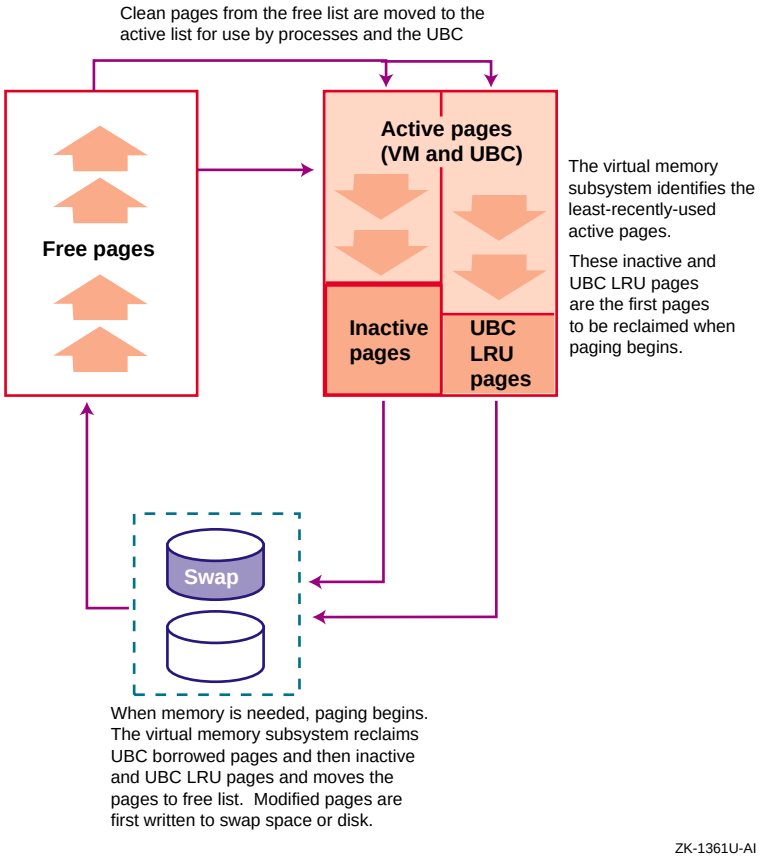
When the memory demand is high and the number of pages on the free page list falls below the value of the vm subsystem attribute vm_page_free_target, the virtual memory subsystem uses paging to replenish the free page list. The page-out daemon and task swapper daemon are extensions of the page reclamation 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 6–7 shows the movement of pages during paging operations.

Figure 6–7: 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 6.1.4.3).

6.1.4.3 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 6.5. 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 6.5.2 and Section 6.5.3 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 6.5.9, Section 6.5.10, Section 6.5.11, and Section 6.5.12 for swap buffer tuning information.

6.2 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 Section 3.6 for information about modifying kernel subsystem attributes.

See Section 2.3.2.2 and Section 2.3.2.3 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 RAID 5 for swap devices. See Chapter 8 for more information about RAID 5.

See the *System Administration* manual for more information about adding swap devices. See Chapter 8 for more information about configuring and tuning disks for high performance and availability.

6.3 Gathering Memory Information

Table 6–2 describes the tools that you can use to gather information about memory usage.

Table 6–2: Virtual Memory and UBC Monitoring Tools

Name	Use	Description
`sys_check`	Analyzes system configuration and displays statistics (Section 4.3)	Creates an HTML file that describes the system configuration, and can be used to diagnose problems. The `sys_check` utility checks kernel variable settings and memory and CPU resources, and provides performance data and lock statistics for SMP systems and kernel profiles. The `sys_check` utility calls various commands and utilities to perform a basic analysis of your configuration and kernel variable settings, and provides warnings and tuning guidelines if necessary. See `sys_check`(8) for more information.
`uerf`	Displays total system memory	Use the `uerf -r 300` command to determine the amount of memory on your system. The beginning of the listing shows the total amount of physical memory (including wired memory) and the amount of available memory. See `uerf`(8) for more information.
`vmstat`	Displays virtual memory and CPU usage statistics (Section 6.3.1)	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.
`ps`	Displays CPU and virtual memory usage by processes (Section 6.3.2)	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.

Table 6–2: Virtual Memory and UBC Monitoring Tools (cont.)

Name	Use	Description
ipcs	Displays IPC statistics	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.
swapon	Displays information about swap space utilization (Section 6.3.3)	Displays the total amount of allocated swap space, swap space in use, and free swap space for each swap device. You can also use the swapon command to allocate additional swap space.
dbx	Reports UBC statistics (Section 6.3.4)	You can check the UBC by using the dbx print command to examine the ufs_getapage_stats data structure, which contains information about UBC page usage.

The following sections describe some of these tools in detail.

6.3.1 Monitoring Memory by Using the vmstat Command

The vmstat command shows the virtual memory, process, and CPU statistics for a specified time interval. The first line of the output is for all time since a reboot, and each subsequent report is for the last interval.

An example of the vmstat command is as follows; output is provided in one-second intervals:

```
# /usr/ucb/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
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 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 6.1.1 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 7.1.2 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 8.2 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 6.3.2 for more information.
- The output of the `swapon` command shows excessive use of swap space. See Section 6.3.3 for more information.

You can also use the `vmstat -P` command to display statistics about physical memory use. For example:

```
# vmstat -P
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 Section 6.4 for information about increasing memory resources.

6.3.2 Monitoring Memory by Using the ps Command

The ps command displays the current status of the system processes. You can use it to determine the current running processes (including users), their state, and how they utilize system memory. The command lists processes in order of decreasing CPU usage, so you can identify which processes are using the most CPU time.

The ps command provides only a snapshot of the system; by the time the command finishes executing, the system state has probably changed. In addition, one of the first lines of the command may refer to the ps command itself.

An example of the ps command is as follows:

```
# /usr/ucb/ps aux
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 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 11.2 for information about decreasing an application's use of memory.
- Are duplicate processes running? Use the kill command to terminate any unnecessary 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 7.2.2) 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(8) 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 6.4 for information about increasing memory resources.

6.3.3 Monitoring Swap Space Usage by Using the swapon Command

Use the swapon -s command to display your swap device configuration. For each swap partition, the command displays the total amount of allocated swap space, the amount of swap space that is being used, and the amount of free swap space. This information can help you determine how your swap space is being utilized.

An example of the swapon command is as follows:

```
# /usr/sbin/swapon -s
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 2.3.2.3 for information about swap space requirements. See Section 6.2 for information about adding swap space and distributing swap space for high performance.

6.3.4 Monitoring the UBC by Using the dbx Debugger

If you have not disabled read-ahead, you can monitor 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
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.

6.4 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 6–3 shows the guidelines for increasing memory resources to processes and lists the performance benefits as well as tradeoffs. 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 6–3: Memory Resource Tuning Guidelines

Guideline	Performance Benefit	Tradeoff
Reduce the number of processes running at the same time (Section 6.4.1)	Decreases CPU load and demand for memory	System performs less work
Reduce the static size of the kernel (Section 6.4.2)	Frees memory	Not all functionality may be available
Decrease the borrowed memory threshold (Section 6.4.3)	Improves system response time when memory is low	May degrade file system performance
Decrease the memory allocated to the AdvFS buffer cache (Section 6.4.4)	Provides more memory resources to processes	May degrade AdvFS performance on systems that open and reuse files

Table 6–3: Memory Resource Tuning Guidelines (cont.)

Guideline	Performance Benefit	Tradeoff
Decrease the memory allocated to AdvFS access-structures (Section 6.4.5)	Provides more memory resources to processes	May degrade AdvFS performance on low-memory systems that use AdvFS
Decrease the size of the metadata buffer cache (Section 6.4.6)	Provides more memory resources to processes	May degrade UFS performance on small systems
Decrease the size of the namei cache (Section 6.4.7)	Frees memory	May slow lookup operations and degrade file system performance
Increase the percentage of memory reserved for kernel `malloc` allocations (Section 6.4.8)	Improves network throughput under a heavy load	Consumes memory
Reduce process memory requirements (Section 11.2.6)	Frees memory	Program may not run optimally

The following sections describe the guidelines that will increase the memory available to processes in detail.

6.4.1 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 the `batch` command to run applications at offpeak hours.

See `at`(1) for more information.

6.4.2 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 Section 3.6 for information about modifying kernel subsystem attributes.

6.4.3 Decreasing the Borrowed Memory Threshold

You may be able to prevent paging by decreasing the borrowed memory threshold. If you do this, less memory remains in the UBC when page reclamation begins. The `ubc_borrowpercent` attribute specifies the UBC

borrowing threshold. See Section 6.1.2.2 for information about borrowed memory.

Performance Benefit and Tradeoff

Decreasing the borrowed memory threshold may improve the system response time when memory is low, but may also reduce UBC effectiveness.

You can modify the `ubc_borrowpercent` attribute without rebooting the system.

When to Tune

If your workload does not use file systems heavily, you may want to decrease the borrowed memory threshold.

Recommended Values

The default value of the `ubc_borrowpercent` attribute is 20 percent.

See Section 3.6 for information about modifying kernel subsystem attributes.

6.4.4 Decreasing the Size of the AdvFS Buffer Cache

The `advfs` subsystem attribute `AdvfsCacheMaxPercent` determines the maximum amount of memory that can be used for the AdvFS buffer cache. See Section 6.1.2.3 for information about the AdvFS buffer cache.

Performance Benefit and Tradeoff

To free memory resources, you may want to decrease the amount of memory allocated to the AdvFS buffer cache. Decreasing the cache size also decreases the overhead associated with managing the cache.

However, decreasing the size of the AdvFS buffer cache may degrade AdvFS performance if you reuse many AdvFS files.

You cannot modify the `AdvfsCacheMaxPercent` attribute without rebooting the system.

When to Tune

If you are not using AdvFS, decrease the size of the AdvFS buffer cache.

If you are using AdvFS, but most of your data is read or written only once, reducing the size of the AdvFS buffer cache may improve performance. The cache lookup time is decreased because the cache contains fewer entries to search. If you are using AdvFS, but you have a large-memory system, you also may want to decrease the size of the AdvFS cache.

Recommended Values

The default value of the AdvfsCacheMaxPercent attribute is 7 percent of physical memory. The minimum is 1 percent; the maximum is 30 percent.

If you are not using AdvFS, decrease the cache size to 1 percent.

See Section 3.6 for information about modifying kernel subsystem attributes.

6.4.5 Decreasing the Memory for AdvFS Access Structures

AdvFS access structures are in-memory data structures that AdvFS uses to cache low-level information about files that are currently open and files that were opened but are now closed. Caching open file information can enhance AdvFS performance if the open files are later reused.

At boot time, the system reserves for AdvFS access structures a portion of the physical memory that is not wired by the kernel. The memory reserved is either twice the value of the AdvfsMinFreeAccess attribute or the value of the AdvfsAccessMaxPercent attribute, whichever is smaller. Access structures are then placed on the access structure free list, and are allocated and deallocated according to the kernel configuration and workload demands.

There are three attributes that control the allocation of AdvFS access structures:

- The AdvfsAccessMaxPercent attribute controls the maximum percentage of pageable memory that can be allocated for AdvFS access structures.
- The AdvfsMinFreeAccess attribute controls the allocation of AdvFS access structures. At boot time, and when the number of access structures on the free list is less than the value of the AdvfsMinFreeAccess attribute, AdvFS allocates access structures, until the number of access structures on the free list is either twice the value of the AdvfsMinFreeAccess attribute or the value of the AdvfsAccessMaxPercent attribute, whichever is smaller.
- The AdvfsMaxFreeAccessPercent attribute controls when access structures are deallocated from the free list. When the percentage of access structures on the free list is more than the value of the AdvfsMaxFreeAccessPercent attribute, and the number of access structures on the free list is more than twice the value of the AdvfsMinFreeAccess attribute, AdvFS deallocates access structures.

See Section 9.3.3 for more information about access structures.

Performance Benefit and Tradeoff

Decreasing the amount of memory allocated for access structures makes more memory available to processes and file system buffer caching, but it may degrade performance on low-memory systems that use AdvFS or systems that reuse AdvFS files.

You can modify the `AdvfsAccessMaxPercent` attribute without rebooting the system.

When to Tune

If you do not use AdvFS, you may want to decrease the value of the `AdvfsMinFreeAccess` attribute to minimize the memory allocated to AdvFS access structures at boot time.

If your workload does not reuse AdvFS files, you may want to decrease the value of the `AdvfsMaxFreeAccessPercent` attribute. This will cause the system to aggressively deallocate free access structures.

If you have a large-memory system, you may want to decrease the value of the `AdvfsAccessMaxPercent` attribute. This is 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.

Recommended Values

The default value of the `AdvfsAccessMaxPercent` attribute is 25 percent of pageable memory. The minumum value is 5 percent; the maximum value is 95 percent.

The default value of the `AdvfsMinFreeAccess` attribute is 128. The minumum value is 1; the maximum value is 100,000.

The default value of the `AdvfsMaxFreeAccessPercent` attribute is 80 percent. The minimum value is 5 percent; the maximum value is 95 percent.

See Section 3.6 for information about modifying kernel subsystem attributes.

6.4.6 Decreasing the Size of the Metadata Buffer Cache

The metadata buffer cache contains recently accessed UFS and CD–ROM File System (CDFS) metadata. The `vfs` subsystem attribute `bufcache` specifies the percentage of physical memory that the kernel wires for the metadata buffer cache.

Performance Benefit and Tradeoff

Decreasing the size of the metadata buffer cache will increase the amount of memory that is available to processes and for file system buffer caching. However, decreasing the size of the cache may degrade UFS performance.

You cannot modify the `bufcache` attribute without rebooting the system.

When to Tune

If you have a high cache hit rate or if you use only AdvFS, you may want to decrease the size of the metadata buffer cache.

Recommended Values

The default size of the metadata buffer cache is 3 percent of physical memory. You can decrease the value of the `bufcache` attribute to a minimum of 1 percent.

For VLM systems and systems that use only AdvFS, set the value of the `bufcache` attribute to 1 percent.

You can reduce the memory allocated for the metadata buffer cache below 1 percent by decreasing the value of the `vfs` subsystem attribute `bufpages`, which specifies the number of pages in the cache. The default value is 1958 pages.

See Section 3.6 for information about modifying kernel subsystem attributes.

6.4.7 Decreasing the Size of the namei Cache

The namei cache is used by UFS, AdvFS, CDFS, and NFS to store information about recently used file names, parent directory vnodes, and file vnodes. The number of vnodes determines the number of open files. The namie cache also stores vnode information for files that were referenced but not found. Having this information in the cache substantially reduces the amount of searching that is needed to perform pathname translations.

The `vfs` subsystem attribute `name_cache_size` specifies the number of elements in the namei cache.

Performance Benefit and Tradeoff

Decreasing the size of the namei cache can free memory resources. However, this may degrade file system performance by reducing the number of cache hits.

You cannot modify the `name_cache_size` attribute without rebooting.

When to Tune

Monitor the namei cache by using the `dbx print` command to examine the `nchstats` data structure. If the hit rate is low, you may want to decrease the cache size. See Section 9.1.2 for information.

Recommended Values

Decrease the number of elements in the namei cache by decreasing the value of the `name_cache_size` attribute. The default value is:

```
2 * (148 + 10 * maxusers) * 11 / 10
```

The maximum value is:

```
2 * max_vnodes * 11 / 10
```

Make sure that decreasing the size of the namei cache does not degrade file system performance.

See Section 3.6 for information about modifying kernel subsystem attributes.

6.4.8 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. To do this, increase the value of the `generic` subsystem attribute `kmemreserve_percent`, which increases the percentage of physical memory reserved for kernel memory allocations that are less than or equal to the page size (8 KB).

Performance Benefit and Tradeoff

Increasing the value of the `kmemreserve_percent` attribute 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 can modify the `kmemreserve_percent` attribute without rebooting.

When to Tune

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.

Recommended Values

The default value of the `kmemreserve_percent` attribute is 0, which means that the percentage of reserved physical memory will be 0.4 percent of available memory or 256, whichever is the smallest value.

Increase the value of the `kmemreserve_percent` attribute (up to the maximum of 75) by small increments until the output of the `vmstat -M` command shows no entries under the `fail_nowait` heading.

See Section 3.6 for information about modifying kernel subsystem attributes.

6.5 Tuning Paging and Swapping Operation

You may be able to improve performance by modifying paging and swapping operations. Very-large memory (VLM) systems should avoid paging and swapping.

Table 6–4 describes the guidelines for controlling paging and swapping and lists the performance benefits and any tradeoffs.

Table 6–4: Paging and Swapping Tuning Guidelines

Guideline	Performance Benefit	Tradeoff
Increase the paging threshold (Section 6.5.1)	Maintains performance when free memory is exhausted	May waste memory
Increase the rate of swapping (Section 6.5.2)	Increases process throughput	Decreases interactive response performance
Decrease the rate of swapping (Section 6.5.3)	Improves process interactive response performance	Decreases process throughput
Enable aggressive swapping (Section 6.5.4)	Improves system throughput	Degrades interactive response performance
Limit process resident set size (Section 6.5.5)	Prevents a process from being swapped out because of a large resident set size	May increase paging activity
Use memory locking (Section 11.2.7)	Prevents a process from being swapped out	Improves process throughput
Increase the rate of dirty page prewriting (Section 6.5.6)	Prevents drastic performance degradation when memory is exhausted	Decreases peak workload performance
Decrease the rate of dirty page prewriting (Section 6.5.7)	Improves peak workload performance	May cause drastic performance degradation when memory is exhausted

Table 6–4: Paging and Swapping Tuning Guidelines (cont.)

Guideline	Performance Benefit	Tradeoff
Increase the size of the page-in and page-out clusters (Section 6.5.8)	Improves peak workload performance	Decreases total system workload performance
Increase the swap device I/O queue depth for page ins and swap outs (Section 6.5.9)	Increases overall system throughput	Consumes memory
Decrease the swap device I/O queue depth for page ins and swap outs (Section 6.5.10)	Improves the interactive response time and frees memory	Decreases system throughput
Increase the swap device I/O queue depth for page outs (Section 6.5.11)	Frees memory and increases throughput	Decreases interactive response performance
Decrease the swap device I/O queue depth for page outs (Section 6.5.12)	Improves interactive response time	Consumes memory

The following sections describe the guidelines for controlling paging and swapping in detail.

6.5.1 Increasing the Paging Threshold

The `vm` subsystem attribute `vm_page_free_target` specifies the minimum number of pages on the free list before paging begins. Increasing the paging threshold may prevent performance problems when a severe memory shortage occurs. See Section 6.1.4 for information about paging and swapping attributes.

Performance Benefit and Tradeoff

Increasing the value of the `vm_page_free_target` attribute (the paging threshold) may improve performance when free memory is exhausted. However, this may increase paging activity on a low-memory system. In addition, an excessively high value can waste memory.

You can modify the `vm_page_free_target` attribute without rebooting the system.

When to Tune

You may 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. Do not increase the value if the system is not paging.

Recommended Values

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

If you want to increase the value of the vm_page_free_target attribute, start at the default value and then double the value. Do not specify a value that is more than 1024 pages or 8 MB.

Do not decrease the value of the vm_page_free_target attribute.

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 Section 3.6 for information about modifying kernel subsystem attributes.

6.5.2 Increasing the Rate of Swapping

Swapping has a drastic impact on system performance. You can modify kernel subsystem attributes to control when swapping begins and ends. VLM systems and systems running large programs should avoid swapping. Hard swapping begins when the number of pages on the free list is less than the value of the vm subsystem attribute vm_page_free_optimal for five seconds. See Section 6.1.4 for more information about paging and swapping attributes.

Performance Benefit and Tradeoff

Increasing the rate of swapping (swapping earlier during page reclamation) by raising 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, so increasing the rate of swapping may degrade interactive response time.

You can modify the vm_page_free_optimal attribute without rebooting the system.

When to Tune

You do not need to modify task swapping if the system is not paging.

Recommended Values

The default value of the vm_page_free_optimal attribute is based on the values of the vm_page_free_target and vm_page_free_min attributes. Increase the value of the vm_page_free_optimal only by 2 pages at a time. Do not specify a value that is more than the value of the vm subsystem attribute vm_page_free_target.

See Section 3.6 for information about modifying kernel subsystem attributes.

6.5.3 Decreasing the Rate of Swapping

Swapping has a drastic impact on system performance. You can modify kernel subsystem attributes to control when swapping begins and ends. VLM systems and systems running large programs should avoid swapping. Hard swapping begins when the number of pages on the free list is less than the value of the vm subsystem attribute vm_page_free_optimal for five seconds. See Section 6.1.4 for more information about paging and swapping attributes.

Performance Benefit and Tradeoff

Decreasing the rate of swapping (swapping later during page reclamation) by decreasing the value of the vm_page_free_optimal attribute improves interactive response time, but at the cost of throughput.

You can modify the vm_page_free_optimal attribute without rebooting the system.

When to Tune

You do not need to modify task swapping if the system is not paging.

Recommended Values

The default value of the vm_page_free_optimal attribute is based on the values of the vm_page_free_target and vm_page_free_min attributes. Decrease the value of the vm_page_free_optimal attribute by 2 pages at a time. Do not specify a value that is less than the value of the vm subsystem attribute vm_page_free_min.

See Section 3.6 for information about modifying kernel attributes.

6.5.4 Enabling Aggressive Task Swapping

Swapping begins when the free page list falls below the swapping threshold, as specified by the vm subsystem attribute vm_page_free_swap. Tasks are swapped in when the demand for memory decreases. You can use the vm subsystem attribute vm_aggressive_swap to enable aggressive task swapping, which causes the virtual memory subsystem to swap in processes at a rate that is slower than normal task swapping.

Performance Benefit and Tradeoff

Aggressive task swapping improves system throughput, but it degrades the interactive response performance.

You can modify the vm_aggressive_swap attribute without rebooting.

When to Tune

Usually, you do not need to enable aggressive task swapping.

Recommended Values

By default, the vm_aggressive_swap attribute is disabled (set to 0). To enable aggressive task swapping, set the value of the vm_aggressive_swap attribute to 1.

See Section 3.6 for information about modifying kernel attributes.

6.5.5 Limiting the Resident Set Size to Avoid Swapping

By default, Tru64 UNIX does not limit the resident set size for a process. If the number of free pages cannot keep up with the demand for memory, processes with large resident set sizes are likely candidates for swapping. To avoid swapping a process because it has a large resident set size, you can specify process-specific and system-wide limits for resident set sizes.

To set a limit on the resident set size for a specific process, use the `setrlimit` system call to specify a value (in bytes) for the `RLIMIT_RSS` resource parameter.

To set a system-wide limit, use the `vm` subsystem attribute `vm_rss_maxpercent`, which specifies the maximum percentage of managed pages that can be used for a resident set.

If you limit the resident set size, either for a specific process or system-wide, 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 system-wide or per-process limit.
- The number of pages of the free page list is less than the value of the `vm` subsystem attribute `vm_rss_block_target`.

A task that has exceeded its soft limit remains blocked until the number of pages on the free page list reaches the value of the `vm` subsystem attribute `vm_rss_wakeup_target` attribute.

Performance Benefit and Tradeoff

Limiting resident set sizes will prevent a process from being swapped out because of a large resident set size.

You cannot modify the `anon_rss_enforce` attribute without rebooting the system. You can modify the `vm_rss_maxpercent`, `vm_rss_block_target`, and `vm_rss_wakeup_target` attributes without rebooting the system.

When to Tune

You do not need to limit resident set sizes if the system is not paging.

Recommended Values

To set a system-wide limit, use the `vm` subsystem attribute `vm_rss_maxpercent` to specify the maximum percentage of managed pages that can be used for a resident set. The minimum value of the attribute is 1; the maximum and default values are 100. Decrease the default value by decrements of 10 percent.

Use the attribute `anon_rss_enforce` to set either a soft or hard limit on the size of a resident set. If set to 0 (zero), the default, there is no limit on

the size of a process' resident set. Set the attribute to 1 to enable a soft limit; set the attribute to 2 to enable a hard limit.

If you enable a soft limit, use the vm_rss_block_target to specify the free page list threshold at which anonymous paging begins. The default value of the vm_rss_block_target attribute is the same as the default value of the vm_page_free_optimal attribute, which specifies the swapping threshold. You can increase the default value of the vm_rss_block_target attribute to delay paging anonymous memory. You can decrease the default value to start paging earlier. The minimum value of the vm_rss_block_target attribute is 0; the maximum value is 2 GB.

If you enable a soft limit, use the vm subsystem attribute vm_rss_wakeup_target attribute to specify the free page list threshold at which a task that has exceeded its soft limit becomes unblocked. The default value of the vm_rss_wakeup_target attribute is the same as the default value of the vm_page_free_optimal attribute, which specifies the swapping threshold. You can increase the value of the vm_rss_wakeup_target attribute to free more memory before unblocking the task. You can decrease the value so that the task is unblocked sooner, but less memory is freed. The minimum value of the vm_rss_block_target attribute is 0; the maximum value is 2 GB.

6.5.6 Increasing Modified Page Prewriting

The virtual memory subsystem attempts to prevent a memory shortage by prewriting modified (dirty) 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.

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.

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 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.

See Section 6.1.4.1 for more information about modified page prewriting.

Performance Benefit and Tradeoff

Increasing the rate of modified page prewriting will prevent a drastic performance degradation when memory is exhausted, but will also reduce peak workload performance. Increasing the rate of modified page prewriting will also increase the amount of continuous disk I/O, but will provide better file system integrity if a system crash occurs.

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 dirty page prewriting if the system is not paging.

Recommended Values

To increase the rate of inactive dirty page prewriting, increase the value of the `vm_page_prewrite_target` attribute by increments of 64 pages. The default value is `vm_page_free_target` * 2.

The default value of the `vm_ubcdirtypercent` attribute 10 percent of the total UBC LRU pages (that is, 10 percent of the total UBC LRU pages must be dirty before the oldest UBC LRU pages are prewritten). To increase the rate of UBC LRU dirty page prewriting, decrease the value of the `vm_ubcdirtypercent` attribute by decrements of 1 percent.

See Section 3.6 for information about modifying kernel attributes.

6.5.7 Decreasing Modified Page Prewriting

The virtual memory subsystem attempts to prevent a memory shortage by prewriting modified (dirty) 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.

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.

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 `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.

See Section 6.1.4.1 for more information about modified page prewriting.

Performance Benefit and Tradeoff

Decreasing the rate of modified page prewriting will improve peak workload performance, but it will cause a drastic performance degradation when memory is exhausted.

You can modify the vm_page_prewrite_target and vm_ubcdirtypercent attributes without rebooting the system.

When to Tune

You do not need to modify inactive dirty page writing if the system is not paging. Decrease UBC LRU dirty page prewriting only for benchmarking.

Recommended Values

To decrease the rate of inactive dirty page prewriting, decrease the default value of the vm_page_prewrite_target attribute. The default value is vm_page_free_target * 2.

The default value of the vm_ubcdirtypercent attribute is 10 percent of the total UBC LRU pages (that is, 10 percent of the UBC LRU pages must be dirty before the UBC LRU pages are prewritten). To decrease the rate of UBC LRU dirty page prewriting, increase the value of the vm_ubcdirtypercent attribute.

See Section 3.6 for information about modifying kernel attributes.

6.5.8 Increasing the Size of the Page-In and Page-Out Clusters

The virtual memory subsystem reads in and writes out additional pages in an attempt to anticipate pages that it will need. The vm subsystem attribute vm_max_rdpgio_kluster specifies the maximum size of an anonymous page-in cluster. The vm subsystem attribute vm_max_wrpgio_kluster specifies the maximum size of an anonymous page-out cluster.

Performance Benefit and Tradeoff

If you increase the value of the vm_max_rdpgio_kluster attribute, the system will spend less time page faulting because more pages will be in memory. This will increase the peak workload performance, but will consume more memory and decrease system performance.

Increasing the value of the vm_max_wrpgio_kluster attribute improves the peak workload performance and conserves memory, but may cause more page ins and decrease the total system workload performance.

You cannot modify the vm_max_rdpgio_kluster and vm_max_wrpgio_kluster attributes without rebooting the system.

When to Tune

You may want to increase the size of the page-in clusters if you have a large-memory system and you are swapping processes. You may want to increase the size of the page-out clusters if you are paging, and you are swapping processes.

Recommended Values

The default value of the `vm_max_rdpgio_kluster` attribute is 16384 bytes (2 pages). The default value of the `vm_max_wrpgio_kluster` attribute is 32768 bytes (4 pages).

See Section 3.6 for information about modifying kernel subsystem attributes.

6.5.9 Increasing the Swap I/O Queue Depth for Page Ins and Swap Outs

Synchronous swap buffers are used for page-in page faults and for swap outs. For each swap device, the `vm` subsystem attribute `vm_syncswapbuffers` specifies the maximum swap device I/O queue depth for page ins and swap outs.

Performance Benefit and Tradeoff:

Increasing the value of the `vm_syncswapbuffers` attribute increases overall system throughput, but it consumes memory.

You can modify the `vm_syncswapbuffers` attribute without rebooting the system.

When to Tune:

Usually, you do not need to decrease the swap I/O queue depth.

Recommended Values:

The default value of the `vm_syncswapbuffers` attribute is 128. The value should be equal to the approximate number of simultaneously running processes that the system can easily handle.

See Section 3.6 for information about modifying kernel subsystem attributes.

6.5.10 Decreasing the Swap I/O Queue Depth for Page Ins and Swap Outs

Synchronous swap buffers are used for page-in page faults and for swap outs. The `vm` subsystem attribute `vm_syncswapbuffers` specifies the maximum swap device I/O queue depth for page ins and swap outs.

Performance Benefit and Tradeoff

Decreasing the value of the vm_syncswapbuffers attribute decreases memory demands and improves interactive response time, but it decreases overall system throughput.

You can modify the vm_syncswapbuffers attribute without rebooting the system.

When to Tune

Usually, you do not have to decrease the swap I/O queue depth.

Recommended Values

The default value of the vm_syncswapbuffers attribute is 128. The value should be equal to the approximate number of simultaneously running processes that the system can easily handle.

See Section 3.6 for information about modifying kernel subsystem attributes.

6.5.11 Increasing the Swap I/O Queue Depth for Page Outs

Asynchronous swap buffers are used for asynchronous page outs and for prewriting modified pages. The vm subsystem attribute vm_asyncswapbuffers controls the maximum depth of the swap device I/O queue for page outs.

Performance Benefit and Tradeoff

Increasing the value of the vm_asyncswapbuffers attribute will free memory and increase the overall system throughput.

You can modify the vm_asyncswapbuffers attribute without rebooting the system.

When to Tune

If you are using LSM, you may want to increase the page-out rate. Be careful if you increase the value of the vm_asyncswapbuffers attribute, because this will cause page-in requests to lag asynchronous page-out requests.

Recommended Values

The default value of the vm_asyncswapbuffers attribute is 4. You can specify a value that is the approximate number of I/O transfers that a swap device can handle at one time.

See Section 3.6 for information about modifying kernel subsystem attributes.

6.5.12 Decreasing the Swap I/O Queue Depth for Page Outs

Asynchronous swap buffers are used for asynchronous page outs and for prewriting modified pages. The vm subsystem attribute vm_asyncswapbuffers controls the maximum depth of the swap device I/O queue for page outs.

Performance Benefit and Tradeoff

Decreasing the vm_asyncswapbuffers attribute will use more memory, but it will improve the interactive response time.

You can modify the vm_asyncswapbuffers attribute without rebooting the system.

When to Tune

Usually, you do not need to decrease the swap I/O queue depth.

Recommended Values

The default value of the vm_asyncswapbuffers attribute is 4. You can specify a value that is the approximate number of I/O transfers that a swap device can handle at one time.

See Section 3.6 for information about modifying kernel subsystem attributes.

6.6 Reserving Physical Memory for Shared Memory

Granularity hints allow you to reserve a portion of dynamically wired 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 6.6.1 and Section 6.6.2 describe how to enable granularity hints.

6.6.1 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 zero.

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, invoke the following sequence of 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 output shows the following:

- 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:

- 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 Section 3.6 for information about modifying kernel subsystem attributes.

6.6.2 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 zero.

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 zero.

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 zero.

7

Managing CPU Performance

You may be able to improve performance by optimizing CPU resources. This chapter describes how to perform the following tasks:

- Obtain information about CPU performance (Section 7.1)
- Improve CPU performance (Section 7.2)

7.1 Gathering CPU Performance Information

Table 7–1 describes the tools you can use to gather information about CPU usage.

Table 7–1: CPU Monitoring Tools

Name	Use	Description
`sys_check`	Analyzes system configuration and displays statistics (Section 4.3)	Creates an HTML file that describes the system configuration, and can be used to diagnose problems. This utility checks kernel variable settings and memory and CPU resources, and provides performance data and lock statistics for SMP systems and kernel profiles. The `sys_check` utility performs a basic analysis of your configuration and kernel variable settings, and provides warnings and tuning guidelines if necessary. See `sys_check`(8) for more information.
`ps`	Displays CPU and virtual memory usage by processes (Section 6.3.2 and Section 7.1.1)	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.

Table 7–1: CPU Monitoring Tools (cont.)

Name	Use	Description
Process Tuner	Displays CPU and virtual memory usage by processes	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.
`vmstat`	Displays virtual memory and CPU usage statistics (Section 7.1.2)	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.
`monitor`	Collects performance data	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. The `monitor` command is available on the Tru64 UNIX Freeware CD-ROM. See `ftp://gatekeeper.dec.com/pub/DEC` for information.
`top`	Provides continuous reports on the system	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`	Displays IPC statistics	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.

Table 7–1: CPU Monitoring Tools (cont.)

Name	Use	Description
`uptime`	Displays the system load average (Section 7.1.3)	Displays 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.
`w`	Reports system load averages and user information	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. The `w` command also displays information about system users, including login and process information. See w(1) for more information.
`xload`	Monitors the system load average	Displays the system load average in a histogram that is periodically updated. See `xload`(1X) for more information.
`(kdbx) cpustat`	Reports CPU statistics (Section 7.1.4)	Displays CPU statistics, including the percentages of time the CPU spends in various states.
`(kdbx) lockstats`	Reports lock statistics (Section 7.1.5)	Displays lock statistics for each lock class on each CPU in the system.

The following sections describe some of these commands in detail.

7.1.1 Monitoring CPU Usage by Using the ps Command

The `ps` command displays a snapshot of the current status of the system processes. You can use it to determine the current running processes (including users), their state, and how they utilize system memory. The command lists processes in order of decreasing CPU usage so you can identify which processes are using the most CPU time.

See Section 6.3.2 for detailed information about using the `ps` command to diagnose CPU performance problems.

7.1.2 Monitoring CPU Statistics by Using the vmstat Command

The `vmstat` command shows the virtual memory, process, and CPU statistics for a specified time interval. The first line of output displays statistics since reboot time; each subsequent line displays statistics since the specified time interval.

An example of the vmstat command is as follows; output is provided in one-second intervals:

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

The following fields are particularly important for CPU monitoring:

- 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.
- 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 6.3.1 for detailed information about the using the vmstat command to diagnose performance problems.

To use the vmstat command to diagnose a CPU performance problem, check the user (us), system (sy), and idle (id) time split. You must understand how your applications use the system to determine the appropriate values for these times. The goal is to keep the CPU as productive as possible. Idle CPU cycles occur when no runnable processes exist or when the CPU is waiting to complete an I/O or memory request.

The following list describes how to interpret the values for user, system, and idle time:

- System time (sy)—A high percentage of system time may indicate a system bottleneck, which can be caused by excessive system calls, device

interrupts, context switches, soft page faults, lock contention, or cache missing.

A high percentage of system time and a low percentage of idle time may indicate that something in the application load is stimulating the system with high overhead operations. Such overhead operations could consist of high system call frequencies, high interrupt rates, large numbers of small I/O transfers, or large numbers of IPCs or network transfers.

A high percentage of system time and low percentage of idle time may also be caused by failing hardware. Use the `uerf` command to check your hardware.

A high percentage of system time may also indicate that the system is thrashing; that is, the amount of memory available to the virtual memory subsystem has gotten so low that the system is spending all its time paging and swapping in an attempt to regain memory. A system that spends more than 50 percent of its time in system mode and idle mode may not have enough memory resources. See Section 6.4 for information about increasing memory resources.

- Idle time (`id`)—A high percentage of idle time on one or more processors indicates either:
  - Threads are blocked because the CPU is waiting for some event or resource (for example, memory or I/O)
  - Threads are idle because the CPU is not busy

  If you have a high idle time and poor response time, and you are sure that your system has a typical load, one or more of the following problems may exist:
  - The hardware may have reached its capacity
  - A kernel data structure is exhausted
  - You may have a memory, disk I/O, or network bottleneck

  If the idle time percentage is very low but performance is acceptable, your system is utilizing its CPU resources efficiently.
- User time (`us`)—A high percentage of user time can be a characteristic of a well-performing system. However, if the system has poor performance, a high percentage of user time may indicate a user code bottleneck, which can be caused by inefficient user code, insufficient CPU processing power, or excessive memory latency or cache missing. See Section 7.2 for information on optimizing CPU resources.

  Use profiling to determine which sections of code consume the most processing time. See Section 11.1 and the *Programmer's Guide* for more information on profiling.

A high percentage of user time and a low percentage of idle time may indicate that your application code is consuming most of the CPU. You can optimize the application, or you may need a more powerful processor. See Section 7.2 for information on optimizing CPU resources.

7.1.3 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 6.3.2 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.

7.1.4 Checking CPU Usage by Using the kdbx Debugger

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.

7.1.5 Checking Lock Usage by Using the kdbx Debugger

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.

7.2 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 may be able to improve performance by optimizing CPU usage.

Table 7–2 describes the guidelines for improving CPU performance.

Table 7–2: Primary CPU Performance Improvement Guidelines

Guideline	Performance Benefit	Tradeoff
Add processors (Section 7.2.1)	Increases CPU resources	Applicable only for multiproccessing systems, and may affect virtual memory performance
Use the Class Scheduler (Section 7.2.2)	Allocates CPU resources to critical applications	None
Prioritize jobs (Section 7.2.3)	Ensures that important applications have the highest priority	None
Schedule jobs at offpeak hours (Section 7.2.4)	Distributes the system load	None
Stop the `advfsd` daemon (Section 7.2.5)	Decreases demand for CPU power	Applicable only if you are not using the AdvFS graphical user interface
Use hardware RAID (Section 7.2.6)	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.

7.2.1 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 7.1.2 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 Prestoserve (see Section 2.4.8), or by using a RAID controller with a write-back cache (see Section 8.5).

7.2.2 Using the Class Scheduler

Use the Class Scheduler to allocate a percentage of CPU time to specific tasks or applications. This allows you to reserve CPU time for important processes, while limiting CPU usage by less critical processes.

To use class scheduling, group together processes into classes and assign each class a percentage of CPU time. You can also manually assign a class to any process.

The Class Scheduler allows you to display statistics on the actual CPU usage for each class.

See the *System Administration* manual and `class_scheduling`(4), `class_admin`(8), `runclass`(1), and `classcntl`(2) for more information about the Class Scheduler.

7.2.3 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.

7.2.4 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.

7.2.5 Stopping the advfsd Daemon

The advfsd daemon allows Simple Network Management Protocol (SNMP) clients such as Netview or Performance Manager (PM) 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, use the following command:

```
# /sbin/init.d/advfsd stop
```

7.2.6 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 8.5 for more information about hardware RAID.

8

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 8.1)
- Managing LSM performance (Section 8.4)
- Managing hardware RAID subsystem performance (Section 8.5)
- Managing Common Access Method (CAM) performance (Section 8.6)

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 2.1, and the guideline's benefits and tradeoffs.

8.1 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.

RAID 0 (data or disk striping) enables you to efficiently distribute data across the disks. See Section 2.5.2 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 (Section 8.4.5). To stripe disks, use a hardware RAID subsystem (Section 8.5).

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 9.3 for more information.

- Use RAID 5.

  RAID 5 distributes disk data and parity data across disks in an array to provide high data availability and to improve read performance. However, RAID 5 decreases write performance in a nonfailure state, and decreases read and write performance in a failure state. RAID 5 can be used for configurations that are mainly read-intensive. As a cost-efficient alternative to mirroring, you can use RAID 5 to improve the availability of rarely-accessed data.

  To create a RAID 5 configuration, use LSM (Section 8.4.6) or a hardware RAID subsystem (Section 8.5).

- 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 9.3.7.4 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 the Logical Storage Manager (LSM) to mirror your swap space. See Section 8.4.2.7 for more information.

  See Section 6.2 for more information about configuring swap devices for high performance.

Section 8.2 describes how to monitor the distribution of disk I/O.

8.2 Monitoring the Distribution of Disk I/O

Table 8–1 describes some commands that you can use to determine if your disk I/O is being distributed.

Table 8–1: Disk I/O Distribution Monitoring Tools

Name	Use	Description
showfdmn	Displays information about AdvFS file domains	Determines if files are evenly distributed across AdvFS volumes. See Section 9.3.5.3 for information.
advfsstat	Displays information about AdvFS file domain and filset usage	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. See Section 9.3.5.1 for information.
swapon	Displays the swap space configuration	Provides information about swap space usage. For each swap partition, the swapon -s command displays the total amount of allocated swap space, the amount of swap space that is being used, and the amount of free swap space. See Section 6.3.3 for information.
volstat	Displays performance statistics for LSM objects	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. See Section 8.4.7.2 for information.
iostat	Displays disk I/O statistics	Provides information about which disks are being used the most. See Section 8.3 for information.

8.3 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 (`dskn`), 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.

8.4 Managing LSM Performance

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.

To obtain the best LSM performance, you must follow the configuration and tuning guidelines described in this manual. The following sections contain:

- Information about LSM features and license requirements (Section 8.4.1)
- Guidelines for disks, disk groups, and databases (Section 8.4.2)
- Guidelines for mirroring volumes (Section 8.4.3)
- Guidelines for using dirty-region logging (DRL) with mirrored volumes (Section 8.4.4)
- Guidelines for striping volumes (Section 8.4.5)
- Guidelines for RAID 5 volumes (Section 8.4.6)
- Information about monitoring the LSM configuration and performance (Section 8.4.7)

See the *Logical Storage Manager* manual for detailed information about using LSM.

8.4.1 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:

- RAID 0 (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.

- RAID 1 (mirroring) maintains copies of data on different disks and reduces the chance that a single disk failure will cause the data to be unavailable.
- RAID 5 (parity RAID) provides data availability through the use of parity data and distributes disk data and file data 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 RAID 5 objects by relocating the affected objects to spare disks or other free space.
- Dirty-region logging (DRL) can be used to improve the recovery time of mirrored volumes after a system failure.
- A graphical user interface (GUI) enables easy disk management and provides detailed performance information.

8.4.2 Basic LSM Disk, Disk Group, and Volume Guidelines

LSM enables you to group disks into storage pools called disk groups. Each disk group maintains a configuration database that contains records describing the LSM objects (volumes, plexes, subdisks, disk media names, and disk access names) that are being used in the disk group.

How you configure your LSM disks, disk groups, and volumes determines the flexibility and performance of your configuration. Table 8–2 describes the LSM disk, disk group, and volume configuration guidelines and lists performance benefits as well as tradeoffs.

Table 8–2: LSM Disk, Disk Group, and Volume Configuration Guidelines

Guideline	Benefit	Tradeoff
Initialize your LSM disks as sliced disks (Section 8.4.2.1)	Uses disk space efficiently	None
Make the `rootdg` disk group a sufficient size (Section 8.4.2.2)	Ensures sufficient space for disk group information	None
Use a sufficient private region size for each disk in a disk group (Section 8.4.2.3)	Ensures sufficient space for database copies	Large private regions require more disk space
Make the private regions in a disk group the same size (Section 8.4.2.4)	Efficiently utilizes the configuration space	None
Organize disk groups according to function (Section 8.4.2.5)	Allows you to move disk groups between systems	Reduces flexibility when configuring volumes

Table 8–2: LSM Disk, Disk Group, and Volume Configuration Guidelines (cont.)

Guideline	Benefit	Tradeoff
Mirror the root file system (Section 8.4.2.6)	Provides availability and improves read performance	Cost of additional disks and small decrease in write performance
Mirror swap devices (Section 8.4.2.7)	Provides availability and improves read performance	Cost of additional disks and small decrease in write performance
Use hot-sparing (Section 8.4.6.3 and Section 8.4.3.5)	Improves recovery time after a disk failure in a mirrored or RAID 5 volume	Requires an additional disk
Save the LSM configuration (Section 8.4.2.8)	Improves availability	None
Use mirrored volumes (Section 8.4.3)	Improves availability and read performance	Cost of additional disks and small decrease in write performance
Use dirty region logging (Section 8.4.4)	Improves resynchronization time after a mirrored volume failure	Slightly increases I/O overhead
Use striped volumes (Section 8.4.5)	Improves performance	Decreases availability
Use RAID 5 volumes (Section 8.4.6)	Provides data availability and improves read performance	Consumes CPU resources, decreases write performance in a nonfailure state, and decreases read and write performance in a failure state

The following sections describe the previous guidelines in detail.

8.4.2.1 Initializing LSM Disks as Sliced Disks

Initialize your LSM disks as sliced disks, instead of configuring individual partitions as simple disks. The disk label for a sliced disk contains information that identifies the partitions containing the private and the public regions. In contrast, simple disks have both public and private regions in the same partition.

A sliced disk places the entire disk under LSM control, uses disk storage efficiently, and avoids using space for multiple private regions on the same disk. When a disk is initialized as an LSM sliced disk, by default, the disk is repartitioned so that partition `g` contains the LSM public region and partition `h` contains the private region. LSM volume data resides in the public region, which uses the majority of the disk starting at block 0. LSM configuration data and metadata reside in the private region, which uses the last 4096 blocks of the disk, by default.

Usually, you do not have to change the size of the LSM private region. See Section 8.4.2.3 for more information.

8.4.2.2 Sizing the rootdg Disk Group

The default disk group, `rootdg` is automatically created when you initialize LSM. Unlike other disk groups, the `rootdg` configuration database contains disk-access records that define all disks under LSM control, in addition to its own disk-group configuration information.

You must make sure that the `rootdg` disk group is large enough to accommodate all the disk-access records. The default size of a configuration database is 4096 blocks. Usually, you do not have to change this value.

8.4.2.3 Sizing Private Regions

LSM keeps the disk media label and configuration database copies in each disk's private region. You must make sure that the private region for each disk is big enough to accommodate the database copies. In addition, the maximum number of LSM objects (disks, subdisks, volumes, and plexes) in a disk group depends on an adequate private region size.

The default private region size is 4096 blocks. Usually, you do not have to modify the default size.

To check the amount of free space in a disk group, use the `voldg list` command and specify the disk group.

You may want to increase the default private region size if you have a very large LSM configuration and need more space for the database copies. Note that a large private region consumes more disk space.

You may want to decrease the default private region size if your LSM configuration is small, and you do not need 4096 blocks for the configuration database. This may improve the LSM startup and disk import times.

Use the `voldisksetup` command with the `privlen` option to set the private region size. See `voldisksetup`(8) for more information.

If you change the size of a disk's private region, all disks that contain the configuration database (that is, if `nconfig` is not 0) should be the same size. See Section 8.4.2.4 for more information.

8.4.2.4 Making Private Regions in a Disk Group the Same Size

The private region of each disk in a disk group should be the same size. This enables LSM to efficiently utilize the configuration database space.

To determine the size of a disk's private region, use the `voldisk list` command and specify the name of the disk.

Use the `voldisksetup` command with the `privlen` option to set the private region size. See `voldisksetup`(8) for more information.

8.4.2.5 Organizing Disk Groups

You may want to organize disk groups according to their function. This enables disk groups to be moved between systems.

Note that using many disk groups decreases the size of the LSM configuration database for each disk group, but it increases management complexity and reduces flexibility when configuring volumes.

8.4.2.6 Mirroring the Root File System

Mirroring the root file system improves overall system availability and also improves read performance for the file system. If a disk containing a copy of the root file system fails, the system can continue running. In addition, if the system is shut down, multiple boot disks can be used to load the operating system and mount the root file system.

Note that mirroring requires additional disks and slightly decreases write performance.

You can configure the root file system under LSM by selecting the option during the full installation, or by encapsulating it into LSM at a later time. The root disk will appear as an LSM volume that you can mirror.

If you mirror the root file system with LSM, you should also mirror the swap devices with LSM. See Section 8.4.2.7 for information about mirroring swap devices.

Note

In a TruCluster configuration, you cannot use LSM to configure the root file system, swap devices, boot partition, quorum

disks, or any partition on a quorum disk. See the TruCluster documentation for more information.

See the *Logical Storage Manager* manual for restrictions and instructions for mirroring the root disk and booting from a mirrored root volume.

8.4.2.7 Mirroring Swap Devices

Mirroring swap devices improves system availability by preventing a system failure caused by a failed swap disk, and also improves read performance. In addition, mirroring both the root file system and swap devices ensures that you can boot the system even if errors occur when you start the swap volume. See Section 8.4.2.6 for information about mirroring the root file system.

Note that mirroring requires additional disks and slightly decreases write performance.

You can configure swap devices under LSM by selecting the option during the full installation or by encapsulating them into LSM at a later time. The swap devices will appear as LSM volumes that you can mirror.

You can also mirror secondary swap devices. Compaq recommends that you use multiple disks for secondary swap devices and add the devices as several individual volumes, instead of striping or concatenating them into a single large volume. This makes the swapping algorithm more efficient.

See the *Logical Storage Manager* manual for restrictions and instructions for mirroring swap space.

8.4.2.8 Saving the LSM Configuration

Use the `volsave` command to periodically create a copy of the LSM configuration. You can use the `volrestore` command to re-create the LSM configuration if you lose a disk group configuration.

See the *Logical Storage Manager* manual for information about saving and restoring the LSM configuration.

8.4.3 LSM Mirrored Volume Configuration Guidelines

Use LSM mirroring (RAID 1) to reduce the chance that a single disk failure will make disk data unavailable. Mirroring maintains multiple copies of volume data on different plexes. If a physical disk that is part of a mirrored plex fails, its plex becomes unavailable, but the system continues to operate using an unaffected plex.

At least two plexes are required to provide data redundancy, and each plex must contain different disks. You can use hot sparing to replace a failed mirrored disk. See Section 8.4.3.5 for information.

Because a mirrored volume has copies of the data on multiple plexes, multiple read operations can be simultaneously performed on the plexes, which dramatically improves read performance. For example, read performance may improve by 100 percent on a mirrored volume with two plexes because twice as many reads can be performed simultaneously. LSM mirroring provides significant I/O read performance benefits with little impact on the CPU.

Writes to a mirrored volume result in simultaneous write requests to each copy of the data, so mirroring may slightly decrease write performance. For example, an individual write request to a mirrored volume may require an additional 5 percent of write time, because the volume write must wait for the completion of the write to each plex.

However, mirroring can improve overall system performance because the read performance that is gained may compensate for the slight decrease in write performance. To determine whether your system performance may benefit from mirroring, use the `volstat` command to compare the number of read operations on a volume to the number of write operations.

Table 8–3 describes LSM mirrored volume configuration guidelines and lists performance benefits as well as tradeoffs.

Table 8–3: LSM Mirrored Volume Guidelines

Guideline	Benefit	Tradeoff
Place mirrored plexes on different disks and buses(Section 8.4.3.1)	Improves performance and increases availability	Cost of additional hardware
Attach multiple plexes to a mirrored volume (Section 8.4.3.2)	Improves performance for read-intensive workloads and increases availability	Cost of additional disks
Use the appropriate read policy (Section 8.4.3.3)	Efficiently distributes reads	None
Use a symmetrical configuration (Section 8.4.3.4)	Provides more predictable performance	None
Configure hot sparing (Section 8.4.3.5)	Increases data availability (highly recommended)	Requires an additional disk device
Use dirty-region logging (DRL) (Section 8.4.4)	Improves mirrored volume recovery rate	May cause an additional decrease in write performance

The following sections describe the previous LSM mirrored volume guidelines in detail.

8.4.3.1 Placing Mirrored Plexes on Different Disks and Buses

Each mirrored plex must contain different disks for effective data redundancy. If you are mirroring a striped plex, each plex must contain different disks for data redundancy. This enables effective striping and mirroring.

By default, the `volassist` command locates plexes so that the loss of a disk will not result in loss of data.

In addition, placing each mirrored plex on a different bus or I/O controller improves performance by distributing the I/O workload and preventing a bottleneck at any one device. Mirroring across different buses also increases availability by protecting against bus and adapter failure.

8.4.3.2 Using Multiple Plexes in a Mirrored Volume

To improve performance for read-intensive workloads, use more than two plexes in a mirrored volume.

Although a maximum of 32 plexes can be attached to the same mirrored volume, using this number of plexes uses disk space inefficiently.

8.4.3.3 Choosing a Read Policy for a Mirrored Volume

To provide optimal performance for different types of mirrored volumes, LSM supports the following read policies:

- `round`

  Reads, in a round-robin manner, from all plexes in the volume.

- `prefer`

  Reads preferentially from the plex that is designated the preferred plex (usually the plex with the highest performance). If one plex exhibits superior performance, either because the plex is striped across multiple disks or because it is located on a much faster device, designate that plex as the preferred plex.

- `select`

  Uses a read policy based on the volume's plex associations. For example, if a mirrored volume contains a single striped plex, that plex is designated the preferred plex. For any other set of plex associations, the `round` policy is used. The `select` read policy is the default policy.

Use the `volprint -t` command to display the read policy for a volume. See Section 8.4.7.1 for information. Use the `volume rdpol` command to set the read policy. See `volprint`(8) and `volume`(8) for information.

8.4.3.4 Using a Symmetrical Plex Configuration

Configure symmetrical plexes for predictable performance and easy management. Use the same number of disks in each mirrored plex. For mirrored striped volumes, you can stripe across half of the available disks to form one plex and across the other half to form the other plex.

In addition, use disks with the same performance characteristics, if possible. You may not gain the performance benefit of a fast disk if it is being used with a slow disk on the same mirrored volume. This is because the overall write performance for a mirrored volume will be determined and limited by the slowest disk. If you have disks with different performance characteristics, group the fast disks into one volume, and group the slow disks in another volume.

8.4.3.5 Using Hot Sparing for Mirrored Volumes

If more than one disk in a mirrored volume fails, you may lose all the data in the volume, unless you configure hot sparing. Compaq recommends that you use LSM hot sparing.

Hot sparing enables you to set up a spare disk that can be automatically used to replace a failed disk in a mirrored set. The automatic replacement capability of hot sparing improves the reliability of mirrored data when a single disk failure occurs.

Note that hot sparing requires an additional disk for the spare disk.

Use the `volwatch -s` command to enable hot sparing. See the *Logical Storage Manager* manual for more information about hot-sparing restrictions and guidelines.

8.4.4 Dirty-Region Logging Configuration Guidelines

For fast resynchronization of a mirrored volume after a system failure, LSM uses dirty-region logging (DRL). However, DRL adds a small I/O overhead for most write access patterns. Typically, the DRL performance degradation is more significant on systems with few writes than on systems with heavy write loads.

DRL logically divides a volume into a set of consecutive regions. Each region is represented by a status bit in the dirty-region log. A write operation to a volume marks the region's status bit as dirty before the data is written to the volume. When a system restarts after a failure, the LSM recovers only those regions of the volume that are marked as dirty in the dirty-region log.

If you disable DRL and the system fails, LSM must copy the full contents of a volume between its mirrors to restore and resynchronize all plexes to a consistent state. Although this process occurs in the background and the volume remains available, it can be a lengthy, I/O-intensive procedure.

Log subdisks are used to store a mirrored volume's dirty-region log. To enable DRL, you must associate at least one log subdisk to a mirrored plex. You can use multiple log subdisks to mirror the log. However, only one log subdisk can exist per plex.

A plex that contains only a log subdisk and no data subdisks is referred to as a log plex. By default, LSM creates a log plex for a mirrored volume. Although you can associate a log subdisk with a regular plex that contains data subdisks, the log subdisk will become unavailable if you detach the plex because one of its data subdisks has failed. Therefore, Compaq recommends that you configure DRL as a log plex.

Table 8–4 describes LSM DRL configuration guidelines and lists performance benefits as well as tradeoffs.

Table 8–4: Dirty-Region Logging Guidelines

Guideline	Benefit	Tradeoff
Configure one log plex for each mirrored volume (Section 8.4.4.1)	Greatly reduces mirror resynchronization time after a system failure.	Slight decrease in write performance
Configure two or more log plexes for each mirrored volume (Section 8.4.4.1)	Greatly reduces mirror resynchronization time after a system failure and provides DRL availability	Slight decrease in write performance

Table 8–4: Dirty-Region Logging Guidelines (cont.)

Guideline	Benefit	Tradeoff
Configure log plexes on disks that are different from the volume's data plexes (Section 8.4.4.1)	Minimizes the logging overhead for writes by ensuring the same disk does not have to seek between the log area and data area for the same volume write	None
Use the default log size (Section 8.4.4.2)	Improves performance	None
Place logging subdisks on infrequently used disks (Section 8.4.4.3)	Helps to prevent disk bottlenecks	None
Use solid-state disks for logging subdisks (Section 8.4.4.4)	Minimizes DRL's write degradation	Cost of solid-state disks
Use a write-back cache for logging subdisks (Section 8.4.4.5)	Minimizes DRL write degradation	Cost of hardware RAID subsystem

The following sections describe the previous DRL guidelines in detail.

8.4.4.1 Configuring Log Plexes

For each mirrored volume, configure one log plex, which is a plex that contains a single log subdisk and no data subdisks. After a system failure, a write to a mirrored volume may have completed on one of its plexes and not on the other plex. LSM must resynchronize each mirrored volume's plex to ensure that all plexes are identical.

A log plex significantly reduces the time it takes to resynchronize a mirrored volume when rebooting after a failure, because only the regions within the volume that were marked as dirty are resynchronized, instead of the entire volume.

By default, LSM creates a log plex for a mirrored volume.

For high availability, you can configure more than one log plex (but only one per plex) for a volume. This ensures that logging can continue even if a disk failure causes one log plex to become unavailable.

In addition, configure multiple log plexes on disks that are different from the volume's data plexes. This will minimize the logging overhead for writes by ensuring that a disk does not have to seek between the log area and data area for the same volume write.

8.4.4.2 Using the Correct Log Size

The size of a dirty-region log is proportional to the volume size and depends on whether you are using LSM in a TruCluster configuration.

For systems not configured as part of a cluster, log subdisks must be configured with two or more sectors. Use an even number, because the last sector in a log subdisk with an odd number of sectors is not used.

The log subdisk size is usually proportional to the volume size. If a volume is less than 2 GB, a log subdisk of two sectors is sufficient. Increase the log subdisk size by two sectors for every additional 2 GB of volume size.

Log subdisks for TruCluster member systems must be configured with 65 or more sectors. Use the same sizing guidelines for non-cluster configurations and multiply that result by 33 to determine the optimal log size for a cluster configuration.

By default, the `volassist addlog` command calculates the optimal log size based on the volume size, so usually you do not have to use the `loglen` attribute to specify a log size. However, the log size that is calculated by default is for a cluster configuration. If a volume will never be used in a cluster, use the `volassist -c addlog` to calculate the optimal log size for a noncluster environment. Compaq recommends that you use the default log size.

See the *Logical Storage Manager* manual for more information about log sizes.

8.4.4.3 Placing Logging Subdisks on Infrequently Used Disks

Place logging subdisks on infrequently used disks. Because these subdisks are frequently written, do not put them on busy disks. In addition, do not configure DRL subdisks on the same disks as the volume data, because this will cause head seeking or thrashing.

8.4.4.4 Using Solid-State Disks for DRL Subdisks

If persistent (nonvolatile) solid-state disks are available, use them for logging subdisks.

8.4.4.5 Using a Nonvolatile Write-Back Cache for DRL

To minimize DRL's impact on write performance, use LSM in conjunction with a RAID subsystem that has a nonvolatile (battery backed) write-back cache. Typically, the DRL performance degradation is more significant on systems with few writes than on systems with heavy write loads.

8.4.5 LSM Striped Volume Configuration Guidelines

Striping data (RAID 0) is useful if you need to write large amounts of data, to quickly read data, or to balance the I/O load from multi-user applications across multiple disks. Striping is especially effective for applications that perform large sequential data transfers or multiple, simultaneous I/O operations. LSM striping provides significant I/O performance benefits with little impact on the CPU.

Striping distributes data in fixed-size units (stripes) across the disks in a volume. Each stripe is a set of contiguous blocks on a disk. The default stripe size (width) is 64 KB. The stripes are interleaved across the striped plex's subdisks, which must be located on different disks to evenly distribute the disk I/O.

The performance benefit of striping depends on the number of disks in the stripe set, the location of the disks, how your users and applications perform I/O, and the width of the stripe. I/O performance improves and scales linearly as you increase the number of disks in a stripe set. For example, striping volume data across two disks can double both read and write performance for that volume (read and write performance improves by 100 percent). Striping data across four disks can improve performance by a factor of four (read and write performance improves by 300 percent).

However, a single disk failure in a volume will make the volume inaccessible, so striping a volume increases the chance that a disk failure will result in a loss of data availability. You can combine mirroring (RAID 1) with striping to obtain high availability. See Section 8.4.3 for mirroring guidelines.

Table 8–5 describes the LSM striped volume configuration guidelines and lists performance benefits as well as tradeoffs.

Table 8–5: LSM Striped Volume Guidelines

Guideline	Benefit	Tradeoff
Use multiple disks in a striped volume (Section 8.4.5.1)	Improves performance by preventing a single disk from being an I/O bottleneck	Decreases volume reliability
Use disks on different buses for the stripe set (Section 8.4.5.2)	Improves performance by preventing a single bus or controller from being an I/O bottleneck	Decreases volume reliability
Use the appropriate stripe width (Section 8.4.5.3)	Ensures that an individual volume I/O is handled efficiently	None

Table 8–5: LSM Striped Volume Guidelines (cont.)

Guideline	Benefit	Tradeoff
Avoid splitting small data transfers (Section 8.4.5.3)	Improves overall throughput and I/O performance by handling small requests efficiently	None
Avoid splitting large data transfers (Section 8.4.5.3)	Improves overall throughput and I/O performance by handling multiple requests efficiently.	Optimizes a volume's overall throughput and performance for multiple I/O requests, instead of for individual I/O requests.

The following sections describe the previous LSM striped volume configuration guidelines in detail.

8.4.5.1 Increasing the Number of Disks in a Striped Volume

Increasing the number of disks in a striped volume can increase the throughput, depending on the applications and file systems you are using and the number of simultaneous users. This helps to prevent a single disk from becoming an I/O bottleneck.

However, a single disk failure in a volume will make the volume inaccessible, so increasing the number of disks in a striped volume reduces the effective mean-time-between-failures (MTBF) of the volume. To provide high availability for a striped volume, you can mirror the striped volume. See Section 8.4.3 for mirroring guidelines.

8.4.5.2 Distributing Striped Volume Disks Across Different Buses

Distribute the disks of a striped volume across different buses or controllers. This helps to prevent a single bus or controller from becoming an I/O bottleneck, but decreases volume reliability.

LSM can obtain I/O throughput and bandwidth that is significantly higher than a hardware RAID subsystem by enabling you to spread the I/O workload for a striped volume across different buses. To prevent a single bus from becoming an I/O bottleneck, configure striped plexes using disks on different buses and controllers, if possible.

You can obtain the best performance benefit by configuring a striped plex so that the stripe columns alternate or rotate across different buses. For example, you could configure a four-way stripe that uses four disks on two

buses so that stripe columns 0 and 2 are on disks located on one bus and stripe columns 1 and 3 are on disks located on the other bus.

However, if you are mirroring a striped volume and you have a limited number of buses, mirroring across buses should take precedence over striping across buses. For example, if you want to configure a volume with a pair of two-way stripes (that is, you want to mirror a two-way stripe) by using four disks on two buses, place one of the plexes of the two-way stripe on disks located on one bus, and configure the other two-way striped plex on the other bus.

For the best possible performance, use a select or round-robin read policy, so that all of the volume's reads and writes will be evenly distributed across both buses. Mirroring data across buses also provides high data availability in case one of the controllers or buses fails.

8.4.5.3 Choosing the Correct LSM Stripe Width

A striped volume consists of a number of equal-sized subdisks, each located on different disks. To obtain the performance benefit of striping, you must select a stripe width that is appropriate for the I/O workload and configuration.

The number of blocks in a stripe unit determines the stripe width. LSM uses a default stripe width of 64 KB (or 128 sectors), which works well in most configurations, such as file system servers or database servers, that perform multiple simultaneous I/Os to a volume. The default stripe width is appropriate for these configurations, regardless of whether the I/O transfer size is small or large.

For highly specialized configurations in which large, raw I/Os are performed one at a time (that is, two or more I/Os are never issued simultaneously to the same volume), you may not want to use the default stripe width. Instead, use a stripe width that enables a large data transfer to be split up and performed in parallel.

The best stripe width for configurations that perform large, individual I/O transfers depends on whether the I/O size varies, the number of disks in the stripe-set, the hardware configuration (for example, the number of available I/O buses), and the disk performance characteristics (for example, average disk seek and transfer times). Therefore, try different stripe widths to determine the width that will provide the best performance for your configuration. Use the LSM online support to obtain help with configuring and deconfiguring plexes with different stripe widths and comparing actual I/O workloads.

If you are striping mirrored volumes, ensure that the stripe width is the same for each plex. Also, avoid striping the same data by both LSM and a

hardware RAID subsystem. If a striped plex is properly configured with LSM, striping the data with hardware RAID may degrade performance.

8.4.6 LSM RAID 5 Configuration Guidelines

RAID 5 provides high availability and improves read performance. A RAID 5 volume contains a single plex, consisting of multiple subdisks from multiple physical disks. Data is distributed across the subdisks, along with parity information that provides data redundancy.

RAID 5 provides data availability through the use of parity, which calculates a value that is used to reconstruct data after a failure. While data is written to a RAID 5 volume, parity is also calculated by performing an exclusive OR (XOR) procedure on the data. The resulting parity information is written to the volume. If a portion of a RAID 5 volume fails, the data that was on that portion of the failed volume is re-created from the remaining data and the parity information.

RAID 5 can be used for configurations that are mainly read-intensive. As a cost-efficient alternative to mirroring, you can use RAID 5 to improve the availability of rarely accessed data.

Notes

LSM mirroring and striping (RAID 0+1) provide significant I/O performance benefits with little impact on the CPU. However, LSM RAID 5 decreases write performance and has a negative impact on CPU performance, because a write to a RAID 5 volume requires CPU resources to calculate the parity information and may involve multiple reads and writes.

In addition, if a disk fails in a RAID 5 volume, write performance will significantly degrade. In this situation, read performance may also degrade, because all disks must be read in order to obtain parity data for the failed disk.

Therefore, Compaq recommends that you use LSM mirroring and striping or hardware (controller-based) RAID, instead of LSM (software-based) RAID 5.

Mirroring RAID 5 volumes and using LSM RAID 5 volumes TruCluster shared storage is not currently supported.

Table 8–6 describes LSM RAID 5 volume configuration guidelines and lists performance benefits as well as tradeoffs. Many of the guidelines for creating striped and mirrored volumes also apply to RAID 5 volumes.

Table 8–6: LSM RAID 5 Volume Guidelines

Guideline	Benefit	Tradeoff
Configure at least one log plex (Section 8.4.6.1)	Increases data availability (highly recommended)	Requires an additional disk
Use the appropriate stripe width (Section 8.4.6.2)	Significantly improves write performance	May slightly reduce read performance
Configure hot sparing (Section 8.4.6.3)	Increases data availability (highly recommended)	Requires an additional disk device

The following sections describe these guidelines in detail.

8.4.6.1 Using RAID 5 Logging

Compaq recommends that you use logging to protect RAID 5 volume data if a disk or system failure occurs. Without logging, it is possible for data not involved in any active writes to be lost or corrupted if a disk and the system fail. If this double failure occurs, there is no way of knowing if the data being written to the data portions of the disks or the parity being written to the parity portions were actually written. Therefore, the recovery of the corrupted disk may be corrupted.

Make sure that each RAID 5 volume has at least one log plex. Do not use a disk that is part of the RAID 5 plex for a log plex.

You can associate a log with a RAID 5 volume by attaching it as an additional, non-RAID 5 layout plex. More than one log plex can exist for each RAID 5 volume, in which case the log areas are mirrored. If you use the `volassist` command to create a RAID 5 volume, a log is created by default.

8.4.6.2 Using the Appropriate Strip Width

Using the appropriate stripe width can significantly improve write performance. However, it may slightly reduce read performance.

The default RAID 5 stripe width is 16 KB, which is appropriate for most environments. To decrease the performance impact of RAID 5 writes, the stripe size used for RAID 5 is usually smaller than the size used for striping (RAID 0).

Unlike striping, splitting a write across all the disks in a RAID 5 set improves write performance, because the system does not have to read existing data to determine the new parity information when it is writing a full striped row of data. For example, writing 64 KB of data to a five-column RAID 5 stripe with a 64-KB stripe width may require two parallel reads, followed by two parallel writes (that is, reads from the existing data and parity information, then writes to the new data and new parity information).

However, writing the same 64 KB of data to a five-column RAID 5 stripe with a 16-KB stripe width may enable the data to be written immediately to disk (that is, five parallel writes to the four data disks and to the parity disk). This is possible because the new parity information for the RAID 5 stripe row can be determined from the 64 KB of data, and reading old data is not necessary.

8.4.6.3 Using Hot Sparing for RAID 5 Volumes

Compaq recommends that you use LSM hot sparing. If more than one disk in a RAID 5 volume fails, you may lose all the data in the volume, unless you configure hot sparing.

Hot sparing enables you to set up a spare disk that can be automatically used to replace a failed RAID 5 disk. The automatic replacement capability of hot sparing improves the reliability of RAID 5 data when a single disk failure occurs. In addition, hot sparing reduces the RAID 5 volume's I/O performance degradation caused by the overhead associated with reconstructing the failed disk's data.

Note that RAID 5 hot sparing requires an additional disk for the spare disk.

Use the `volwatch -s` command to enable hot sparing. See the *Logical Storage Manager* manual for more information about hot-sparing restrictions and guidelines.

8.4.7 Gathering LSM Information

Table 8–7 describes the tools you can use to obtain information about the LSM.

Table 8–7: LSM Monitoring Tools

Name	Use	Description
`volprint`	Displays LSM configuration information (Section 8.4.7.1)	Displays information about LSM disk groups, disk media, volumes, plexes, and subdisk records. It does not display disk access records. See `volprint`(8) for more information.

Table 8–7: LSM Monitoring Tools (cont.)

Name	Use	Description
volstat	Monitors LSM performance statistics (Section 8.4.7.2)	For LSM volumes, plexes, subdisks, or disks, displays either the total performance statistics since the statistics were last reset (or the system was booted), or the current performance statistics within a specified time interval. These statistics include information about read and write operations, including the total number of operations, the number of failed operations, the number of blocks read or written, and the average time spent on the operation. The volstat utility also can reset the I/O statistics. See volstat(8) for more information.
voltrace	Tracks LSM operations (Section 8.4.7.3)	Sets I/O tracing masks against one or all volumes in the LSM configuration and logs the results to the LSM default event log, /dev/volevent. The utility also formats and displays the tracing mask information and can trace the following ongoing LSM events: requests to logical volumes, requests that LSM passes to the underlying block device drivers, and I/O events, errors, and recoveries. See voltrace(8) for more information.
volwatch	Monitors LSM events (Section 8.4.7.4)	Monitors LSM for failures in disks, volumes, and plexes, and sends mail if a failure occurs. The volwatch script automatically starts when you install LSM. The script also enables hot sparing. See volwatch(8) for more information.
volnotify	Monitors LSM events (Section 8.4.7.5)	Displays events related to disk and configuration changes, as managed by the LSM configuration daemon, vold. The volnotify utility displays requested event types until killed by a signal, until a given number of events have been received, or until a given number of seconds have passed. See volnotify(8) for more information.

Note

In a TruCluster configuration, the volstat, voltrace, and volnotify tools provide information only for the member system on which you invoke the command. Use Event Manager, instead of the volnotify utility, to obtain information about LSM

events from any cluster member system. See EVM(5) for more information.

The following sections describe some of these commands in detail.

8.4.7.1 Displaying Configuration Information by Using the volprint Utility

The volprint utility displays information about LSM objects (disks, subdisks, disk groups, plexes, and volumes). You can select the objects (records) to be displayed by name or by using special search expressions. In addition, you can display record association hierarchies, so that the structure of records is more apparent. For example, you can obtain information about failed disks in a RAID 5 configuration, I/O failures, and stale data.

Invoke the voldisk list command to check disk status and display disk access records or physical disk information.

The following example uses the volprint utility to show the status of the voldev1 volume:

```
# /usr/sbin/volprint -ht voldev1
Disk group: rootdg

V  NAME       USETYPE    KSTATE   STATE    LENGTH  READPOL   PREFPLEX
PL NAME       VOLUME     KSTATE   STATE    LENGTH  LAYOUT    NCOL/WID MODE
SD NAME       PLEX       DISK     DISKOFFS LENGTH  [COL/]OFF DEVICE   MODE

v  voldev1    fsgen      ENABLED  ACTIVE   209712  SELECT    -
pl voldev1-01 voldev1    ENABLED  ACTIVE   209712  CONCAT    -        RW
sd dsk2-01    voldev1-01 dsk2     65       209712  0         dsk2     ENA
pl voldev1-02 voldev1    ENABLED  ACTIVE   209712  CONCAT    -        RW
sd dsk3-01    voldev1-02 dsk3     0        209712  0         dsk3     ENA
pl voldev1-03 voldev1    ENABLED  ACTIVE   LOGONLY CONCAT    -        RW
sd dsk2-02    voldev1-03 dsk2     0        65      LOG       dsk2     ENA
```

The following volprint command output shows that the RAID 5 volume r5vol is in degraded mode:

```
# volprint -ht
V    NAME     USETYPE KSTATE  STATE    LENGTH  READPOL  PREFPLEX
PL   NAME     VOLUME  KSTATE  STATE    LENGTH  LAYOUT   NCOL/WID   MODE
SD   NAME     PLEX    DISK    DISKOFFS         LENGTH   [COL/]OFF  DEVICE MODE

v    r5vol    RAID5   ENABLED DEGRADED         20480    RAID  -
pl   r5vol-01         r5vol   ENABLED  ACTIVE  20480    RAID  3/16 RW
sd   disk00-00        r5vol-01         disk00  0        10240 0/0  dsk4d1
sd   disk01-00        r5vol-01         disk01  0        10240 1/0  dsk2d1 dS
sd   disk02-00        r5vol-01         disk02  0        10240 2/0  dsk3d1 -
pl   r5vol-l1         r5vol   ENABLED  LOG     1024     CONCAT -   RW
sd   disk03-01        r5vol-l1         disk00  10240    1024   0   dsk3d0 -
pl   r5vol-l2         r5vol   ENABLED  LOG     1024     CONCAT -   RW
sd   disk04-01        r5vol-l2         disk02  10240    1024   0   dsk1d1 -
```

The output shows that volume r5vol is in degraded mode, as shown by the STATE, which is listed as DEGRADED. The failed subdisk is disk01-00,

as shown by the last column, where the d indicates that the subdisk is detached, and the S indicates that the subdisk contents are stale.

It is also possible that a disk containing a RAID 5 log could experience a failure. This has no direct effect on the operation of the volume; however, the loss of all RAID 5 logs on a volume makes the volume vulnerable to a complete failure.

The following volprint command output shows a failure within a RAID 5 log plex:

```
# volprint -ht
V    NAME     USETYPE KSTATE  STATE    LENGTH  READPOL  PREFPLEX
PL   NAME     VOLUME  KSTATE  STATE    LENGTH  LAYOUT   NCOL/WID   MODE
SD   NAME     PLEX    DISK    DISKOFFS         LENGTH   [COL/]OFF  DEVICE MODE
v    r5vol    RAID5   ENABLED ACTIVE   20480   RAID -
pl   r5vol-01         r5vol   ENABLED  ACTIVE  20480    RAID  3/16 RW
sd   disk00-00        r5vol-01         disk00  0        10240 0/0  dsk4d1 ENA
sd   disk01-00        r5vol-01         disk01  0        10240 1/0  dsk2d1 dS
sd   disk02-00        r5vol-01         disk02  0        10240 2/0  dsk3d1 ENA
pl   r5vol-l1         r5vol  DISABLED          BADLOG   1024  CONCAT -    RW
sd   disk03-01        r5vol-l1         disk00  10240    1024  0    dsk3d0 ENA
pl   r5vol-l2         r5vol  ENABLED   LOG     1024     CONCAT -          RW
sd   disk04-01        r5vol-l2         disk02  10240    1024  0    dsk1d1 ENA
```

The previous command output shows that the RAID 5 log plex r5vol-l1 has failed, as indicated by the BADLOG plex state.

See volprint(8) for more information.

8.4.7.2 Monitoring Performance Statistics by Using the volstat Utility

The volstat utility provides information about activity on volumes, plexes, subdisks, and disks under LSM control. It reports statistics that reflect the activity levels of LSM objects since boot time.

In a TruCluster configuration, the volstat utility provides information only for the member system on which you invoke the command.

The amount of information displayed depends on which options you specify with the volstat utility. For example, you can display statistics for a specific LSM object, or you can display statistics for all objects at one time. If you specify a disk group, only statistics for objects in that disk group are displayed. If you do not specify a particular disk group, the volstat utility displays statistics for the default disk group (rootdg).

You can also use the volstat utility to reset the base statistics to zero. This can be done for all objects or only for specified objects. Resetting the statistics to zero before a particular operation makes it possible to measure the subsequent impact of that operation.

LSM records the following three I/O statistics:

- A count of read and write operations.

- The number of read and write blocks.
- The average operation time. This time refects the total time it took to complete an I/O operation, including the time spent waiting in a disk queue on a busy device.

LSM records these statistics for logical I/Os for each volume. The statistics are recorded for the following types of operations: reads, writes, atomic copies, verified reads, verified writes, plex reads, and plex writes. For example, one write to a two-plex volume requires updating statistics for the volume, both plexes, one or more subdisks for each plex, and one disk for each subdisk. Likewise, one read that spans two subdisks requires updating statistics for the volume, both subdisks, and both disks that contain the subdisks.

Because LSM maintains various statistics for each disk I/O, you can use LSM to understand your application's I/O workload and to identify bottlenecks. LSM often uses a single disk for multiple purposes to distribute the overall I/O workload and optimize I/O performance. If you use traditional disk partitions, monitoring tools combine statistics for an entire disk. If you use LSM, you can obtain statistics for an entire disk and also for its subdisks, which enables you to determine how the disk is being used (for example, by file system operations, raw I/O, swapping, or a database application).

LSM volume statistics enable you to characterize the I/O usage pattern for an application or file system, and LSM plex statistics can determine the effectiveness of a striped plex's stripe width (size). You can also combine LSM performance statistics with the LSM online configuration support tool to identify and eliminate I/O bottlenecks without shutting down the system or interrupting access to disk storage.

After measuring actual data-access patterns, you can adjust the placement of file systems. You can reassign data to specific disks to balance the I/O load among the available storage devices. You can reconfigure volumes on line after performance patterns have been established without adversely affecting volume availability.

LSM also maintains other statistical data. For example, read and write failures that appear for each mirror, and corrected read and write failures for each volume, accompany the read and write failures that are recorded.

The following example displays statical data for volumes:

```
# volstat -vpsd
                  OPERATIONS          BLOCKS          AVG TIME(ms)
TYP NAME        READ     WRITE     READ     WRITE     READ     WRITE
dm  dsk6           3        82       40     62561      8.9      51.2
dm  dsk7           0       725        0    176464      0.0      16.3
dm  dsk9         688        37   175872       592      3.9       9.2
dm  dsk10      29962         0  7670016         0      4.0       0.0
dm  dsk12          0     29962        0   7670016     0.0      17.8
```

```
vol v1            3        72      40     62541    8.9     56.5
pl  v1-01         3        72      40     62541    8.9     56.5
sd  dsk6-01       3        72      40     62541    8.9     56.5
vol v2            0        37       0       592    0.0     10.5
pl  v2-01         0        37       0       592    0.0      8.0
sd  dsk7-01       0        37       0       592    0.0      8.0
sd  dsk12-01      0         0       0         0    0.0      0.0
pl  v2-02         0        37       0       592    0.0      9.2
sd  dsk9-01       0        37       0       592    0.0      9.2
sd  dsk10-01      0         0       0         0    0.0      0.0
pl  v2-03         0         6       0        12    0.0     13.3
sd  dsk6-02       0         6       0        12    0.0     13.3
```

See volstat(8) for more information.

8.4.7.3 Tracking Operations by Using the voltrace Utility

Use the voltrace utility to trace operations on volumes. You can set I/O tracing masks against a group of volumes or the entire system. You can then use the voltrace utility to display ongoing I/O operations relative to the masks.

In a TruCluster configuration, the voltrace utility provides information only for the member system on which you invoke the command.

The trace records for each physical I/O show a volume and buffer-pointer combination that enables you to track each operation, even though the traces may be interspersed with other operations. Similar to the I/O statistics for a volume, the I/O trace statistics include records for each physical I/O done, and a logical record that summarizes all physical records.

Note

Because the voltrace requires significant overhead and produces a large output, run the command only occasionally.

The following example uses the voltrace utility to trace volumes:

```
# /usr/sbin/voltrace -l
96 598519 START read vdev v2 dg rootdg dev 40,6 block 89 len 1 concurrency 1 pid 43
96 598519 END read vdev v2 dg rootdg op 926159 block 89 len 1 time 1
96 598519 START read vdev v2 dg rootdg dev 40,6 block 90 len 1 concurrency 1 pid 43
96 598519 END read vdev v2 dg rootdg op 926160 block 90 len 1 time 1
```

See voltrace(8) for more information.

8.4.7.4 Monitoring Events by Using the volwatch Script

The volwatch script is automatically started when you install LSM. This script sends mail if certain LSM configuration events occur, such as a plex detach caused by a disk failure. The script also enables hot sparing.

The volwatch script sends mail to root by default. To specify another mail recipient or multiple mail recipients, use the rcmgr command to set the rc.config.common variable VOLWATCH_USERS.

See volwatch(8) for more information.

8.4.7.5 Monitoring Events by Using the volnotify Utility

The volnotify utility monitors events related to disk and configuration changes, as managed by the vold configuration daemon. The volnotify utility displays requested event types until killed by a signal, until a given number of events have been received, or until a given number of seconds have passed.

The volnotify utility can display the following events:

- Disk group import, deport, and disable events
- Plex, volume, and disk detach events
- Disk change events
- Disk group change events

In a TruCluster configuration, the volnotify utility only reports events that occur locally on the member system. Therefore, use EVM to get LSM events that occur anywhere within the cluster.

8.5 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. 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 8.5.1)
- Hardware RAID products (Section 8.5.2)
- Guidelines for hardware RAID configurations (Section 8.5.3)

See the hardware RAID product documentation for detailed configuration information.

8.5.1 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. 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 be battery-backed to protect against data loss and corruption.

- RAID support

  All hardware RAID subsystems support RAID 0 (disk striping), RAID 1 (disk mirroring), and RAID 5. High-performance RAID array subsystems also support RAID 3 and dynamic parity RAID. See Section 1.2.3.1 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

8.5.2 Hardware RAID Products

There are different types of hardware RAID subsystems, which provide various degrees of performance and availability at various costs. Compaq 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 an I/O bus slot, either a PCI bus slot or an EISA 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 HSZ50 controller, is connected to a system through a FWD 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 battery-backed write-back cache, and components that can be hotswapped.

- Enterprise Storage Arrays (ESA)

  These preconfigured high-performance hardware RAID subsystems, such as the RAID Array 10000, 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 hotswapped; a standard battery-backed write-back cache; and centralized storage management.

See the Compaq *Systems & Options Catalog* for detailed information about hardware RAID subsystem features.

8.5.3 Hardware RAID Configuration Guidelines

Table 8–8 describes the hardware RAID subsystem configuration guidelines and lists performance benefits as well as tradeoffs.

Table 8–8: Hardware RAID Subsystem Configuration Guidelines

Guideline	Performance Benefit	Tradeoff
Evenly distribute disks in a storage set across different buses (Section 8.5.3.1)	Improves performance and helps to prevent bottlenecks	None
Use disks with the same data capacity in each storage set (Section 8.5.3.2)	Simplifies storage management	None
Use an appropriate stripe size (Section 8.5.3.3)	Improves performance	None
Mirror striped sets (Section 8.5.3.4)	Provides availability and distributes disk I/O performance	Increases configuration complexity and may decrease write performance
Use a write-back cache (Section 8.5.3.5)	Improves write performance, especially for RAID 5 storage sets	Cost of hardware
Use dual-redundant RAID controllers (Section 8.5.3.6)	Improves performance, increases availability, and prevents I/O bus bottlenecks	Cost of hardware
Install spare disks (Section 8.5.3.7)	Improves availability	Cost of disks
Replace failed disks promptly (Section 8.5.3.7)	Improves performance	None

The following sections describe some of these guidelines. See your RAID subsystem documentation for detailed configuration information.

8.5.3.1 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.

8.5.3.2 Using Disks with the Same Data Capacity

Use disks with the same capacity in a storage set. This simplifies storage management.

8.5.3.3 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 2.1 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 eight data transfers.

In addition, the following guidelines can help you to 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.

8.5.3.4 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.

8.5.3.5 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 RAID 5 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 battery-backed to protect against data loss and corruption.

If you are using an HSZ40 or HSZ50 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.

8.5.3.6 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.

8.5.3.7 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.

8.6 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 zero, 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.

9

Managing File System Performance

The Tru64 UNIX operating system supports different file system options that have various performance features and functionality.

This chapter describes the following:

- Gathering information about all types of file systems (Section 9.1)
- Applying tuning guidelines that are applicable to all types of file systems (Section 9.2)
- Managing Advanced File System (AdvFS) performance (Section 9.3)
- Managing UNIX File System (UFS) performance (Section 9.4)
- Managing Network File System (NFS) performance (Section 9.5)

9.1 Gathering File System Information

The following sections describe how to use tools to monitor general file system activity and describe some general file system tuning guidelines.

See Section 6.3.4 for information about using `dbx` to check the Unified Buffer Cache (UBC).

9.1.1 Displaying File System Disk Space

The `df` command displays the disk space used by a UFS file system or AdvFS fileset. Because an AdvFS fileset can use multiple volumes, the `df` command reflects disk space usage somewhat differently than UFS.

For example:

```
# df /usr/var/spool/mqueue
Filesystem   512-blocks        Used   Available Capacity  Mounted on
/dev/rz13e      2368726         882     2130970     1%    /usr/var/spool/mqueue

# df /usr/sde
Filesystem     512-blocks        Used   Available Capacity  Mounted on
flume_sde#sde     1048576      319642      709904    32%    /usr/sde
```

See `df`(1) for more information.

9.1.2 Checking the namei Cache with the dbx Debugger

The namei cache is used by UNIX File System (UFS), Advanced File System (AdvFS), CD–ROM File System (CDFS), Memory File System (MFS), and Network File System (NFS) to store information about recently used file names, parent directory vnodes, and file vnodes. The number of vnodes determines the number of open files. The namei cache also stores vnode information for files that were referenced but not found. Having this information in the cache substantially reduces the amount of searching that is needed to perform pathname translations.

To check namei cache statistics, use the `dbx print` command and specify a processor number to examine the `nchstats` data structure. Consider the following example:

```
# /usr/ucb/dbx –k /vmunix /dev/mem
(dbx) print processor_ptr[0].nchstats
struct {
   ncs_goodhits = 47967479
    ncs_neghits = 3611935
    ncs_badhits = 1828974
    ncs_falsehits = 58393
    ncs_miss = 4194525
    ncs_long = 60
    ncs_badtimehits = 406034
    ncs_collisions = 149
    ncs_unequaldups = 0
    ncs_pad = {
        [0] 0
        [1] 0
        [2] 0
    }
}
(dbx)
```

Examine the `ncs_goodhits` (found a match), `ncs_neghits` (found a match that did not exist), and `ncs_miss` (did not find a match) fields to determine the hit rate. The hit rate should be above 80 percent (`ncs_goodhits` plus `ncs_neghits` divided by the sum of the `ncs_goodhits`, `ncs_neghits`, `ncs_miss`, and `ncs_falsehits` fields). See Section 9.2.1 for information on how to improve the namei cache hit rate and lookup speeds.

If the value in the `ncs_badtimehits` field is more than 0.1 percent of the `ncs_goodhits` field, then you may want to delay vnode deallocation. See Section 9.2.2 for more information.

9.2 Tuning File Systems

You may be able to improve I/O performance by modifying some kernel subsystem attributes that affect file system performance. General file system tuning often involves tuning the Virtual File System (VFS), which provides a uniform interface that allows common access to files, regardless of the file system on which the files reside.

To successfully improve file system performance, you must understand how your applications and users perform disk I/O, as described in Section 2.1. Because file systems share memory with processes, you should also understand virtual memory operation, as described in Chapter 6.

Table 9–1 describes the guidelines for general file system tuning and lists the performance benefits as well as the tradeoffs. There are also specific guidelines for AdvFS and UFS file systems. See Section 9.3 and Section 9.4 for information.

Table 9–1: General File System Tuning Guidelines

Guideline	Performance Benefit	Tradeoff
Increase the size of the namei cache (Section 9.2.1)	Improves namei cache lookup operations	Consumes memory
Delay vnode deallocation (Section 9.2.2)	Improves namei cache lookup operations	Consumes memory
Delay vnode recycling (Section 9.2.3)	Improves cache lookup operations	None
Increase the memory allocated to the UBC (Section 9.2.4)	Improves file system I/O performance	May cause excessive paging and swapping
Decrease the amount of memory borrowed by the UBC (Section 9.2.5)	Improves file system I/O performance	Decreases the memory available for processes, and may decrease system response time
Increase the minimum size of the UBC (Section 9.2.6)	Improves file system I/O performance	Decreases the memory available for processes
Increase the amount of UBC memory used to cache a large file (Section 9.2.7)	Improves large file performance	May allow a large file to consume all the pages on the free list
Disable flushing file read access times (Section 9.2.8)	Improves file system performance for systems that perform mainly read operations	Jeopardizes the integrity of read access time updates and violates POSIX standards
Use Prestoserve to cache only file system metadata (Section 9.2.9)	Improves performance for applications that access large amounts of file system metadata	Prestoserve is not supported in a cluster or for nonfile system I/O operations

The following sections describe these guidelines in detail.

9.2.1 Increasing the Size of the namei Cache

The namei cache is used by UFS, AdvFS, CDFS, and NFS to store information about recently used file names, parent directory vnodes, and file vnodes. The number of vnodes determines the number of open files. The namei cache also stores vnode information for files that were referenced but not found. Having this information in the cache substantially reduces the amount of searching that is needed to perform pathname translations.

The `vfs` subsystem attribute `name_cache_size` specifies the maximum number of elements in the cache. You can also control the size of the namei cache with the `maxusers` attribute, as described in Section 5.1.

Performance Benefit and Tradeoff

You may be able to make lookup operations faster by increasing the size of the namei cache. However, this increases the amount of wired memory.

Note that many benchmarks perform better with a large namei cache.

You cannot modify the `name_cache_size` attribute without rebooting the system.

When to Tune

Monitor the namei cache by using the `dbx print` command and specifying a processor number to examine the `nchstats` data structure. If the miss rate (misses / (good + negative + misses)) is more than 20 percent, you may want to increase the cache size. See Section 9.1.2 for more information.

Recommended Values

The default value of the vfs subsystem attribute name_cache_size is:

```
2 * (148 + 10 * maxusers) * 11 / 10
```

See Section 3.6 for information about modifying kernel subsystem attributes.

9.2.2 Delaying vnode Deallocation

File systems use a kernel data structure called a vnode for each open file. The number of vnodes determines the number of open files. By default, Tru64 UNIX uses dynamic vnode allocation, which enables the supply of kernel vnodes to increase and decrease, according to the system demand.

You enable and disable dynamic vnode allocation by using the vfs subsystem attribute vnode_deallocation_enable, which is set to 1 (enabled), by default. If you disable dynamic vnode allocation, the operating system will use a static vnode pool. For the best performance, Compaq recommends that you use dynamic vnode allocation.

If you are using dynamic vnode allocation, a vnode is deallocated (removed from the free list and its memory is returned to the system) when it has not been accessed through the namei cache for more than the amount of time specified by the vfs subsystem attribute namei_cache_valid_time. The default value is 1200 seconds.

Performance Benefit and Tradeoff

Increasing the default value of the namei_cache_valid_time attribute delays vnode deallocation, which may improve the cache hit rate. However, this will increase the amount of memory consumed by the vnode pool.

You cannot modify the namei_cache_valid_time attribute without rebooting the system.

When to Tune

The default value of the namei_cache_valid_time attribute (1200 seconds) is appropriate for most workloads. However, for workloads with heavy vnode pool activity, you may be able to optimize performance by modifying the default value.

You can obtain namei cache statistics for the number of cache lookup failures due to vnode deallocation by examining the ncs_badtimehits field in the dbx nchstats data structure. If the value in the ncs_badtimehits field is more than 0.1 percent of the successful cache hits, as specified in the ncs_goodhits field, then you may want to increase the default value of the namei_cache_valid_time attribute. See Section 9.1.2 for more information about monitoring the namei cache.

Recommended Values

To delay the deallocation of vnodes, increase the value of the `vfs` subsystem attribute `namei_cache_valid_time`. The default value is 1200.

Note

Decreasing the value of the `namei_cache_valid_time` attribute accelerates the deallocation of vnodes from the namei cache and reduces the efficiency of the cache.

See Section 3.6 for information about modifying kernel subsystem attributes.

9.2.3 Delaying vnode Recycling

File systems use a kernel data structure called a vnode for each open file. The number of vnodes determines the number of open files. By default, Tru64 UNIX uses dynamic vnode allocation, which enables the supply of kernel vnodes to increase and decrease, according to the system demand.

You enable and disable dynamic vnode allocation by using the `vfs` subsystem attribute `vnode_deallocation_enable`, which is set to 1 (enabled), by default. If you disable dynamic vnode allocation, the operating system will use a static vnode pool. For the best performance, Compaq recommends that you use dynamic vnode allocation.

Using dynamic vnode allocation, a vnode can be recycled and used to represent a different file object when it has been on the vnode free list for more than the amount of time specified by the `vfs` subsystem attribute `vnode_age`. The default value is 120 seconds.

Performance Benefit and Tradeoff

Increasing the value of the `vnode_age` attribute delays vnode recycling and increases the chance of a cache hit. However, delaying vnode recycling increases the length of the free list and the amount of memory consumed by the vnode pool.

You can modify the `vnode_age` attribute without rebooting the system.

When to Tune

The default value of the `vnode_age` attribute is appropriate for most workloads. However, for workloads with heavy vnode pool activity, you may be able to optimize performance by modifying the default value.

Recommended Values

To delay the recycling of vnodes, increase the default value of the vnode_age attribute. The default value is 120 seconds.

Decreasing the value of the vnode_age attribute accelerates vnode recycling, but decreases the chance of a cache hit.

See Section 3.6 for information about modifying kernel subsystem attributes.

9.2.4 Increasing Memory for the UBC

The Unified Buffer Cache (UBC) shares with processes the memory that is not wired. The UBC caches UFS and CDFS file system data for reads and writes, AdvFS metadata and file data, and MFS data. Performance is improved if the cached data is later reused and a disk operation is avoided.

The vm subsystem attribute ubc_maxpercent specifies the maximum amount of nonwired memory that can be allocated to the UBC. See Section 6.1.2.2 for information about UBC memory allocation.

Performance Benefit and Tradeoff

If you reuse data, increasing the size of the UBC will improve the chance that data will be found in the cache. An insufficient amount of memory allocated to the UBC can impair file system performance. However, the performance of an application that generates a lot of random I/O will not be improved by a large UBC, because the next access location for random I/O cannot be predetermined.

Be sure that allocating more memory to the UBC does not cause excessive paging and swapping.

You can modify the ubc_maxpercent attribute without rebooting the system.

When to Tune

For most configurations, use the default value of the ubc_maxpercent attribute (100 percent).

Recommended Values

To increase the maximum amount of memory allocated to the UBC, you can increase the value of the vm subsystem attribute ubc_maxpercent. The default value is 100 percent, which should be appropriate for most configurations, including Internet servers.

See Section 3.6 for information about modifying kernel subsystem attributes.

9.2.5 Increasing the Borrowed Memory Threshold

The UBC borrows all physical memory between the value of the vm subsystem attribute ubc_borrowpercent and the value of the ubc_maxpercent attribute. See Section 6.1.2.2 for more information about allocating memory to the UBC.

Performance Benefit and Tradeoff

Increasing the value of the ubc_borrowpercent attribute will reduce the amount of memory that the UBC borrows from processes and allow more memory to remain in the UBC when page reclamation begins. This can increase the UBC cache effectiveness, but it may degrade system response time when a low-memory condition occurs (for example, a large process working set).

You can modify the ubc_borrowpercent attribute without rebooting the system.

When to Tune

If vmstat output shows excessive paging but few or no page outs, you may want to increase the borrowing threshold.

Recommended Values

The value of the ubc_borrowpercent attribute can range from 0 to 100. The default value is 20 percent.

See Section 3.6 for information about modifying kernel subsystem attributes.

9.2.6 Increasing the Minimum Size of the UBC

The minimum amount of memory that can be allocated to the UBC is specified by the vm subsystem attribute ubc_minpercent. See Section 6.1.2.2 for information about allocating memory to the UBC.

Performance Benefit and Tradeoff

Increasing the minimum size of the UBC will prevent large programs from completely consuming the memory that can be used by the UBC.

Because the UBC and processes share virtual memory, increasing the minimum size of the UBC may cause the system to page.

You can modify the ubc_minpercent attribute without rebooting the system.

When to Tune

For I/O servers, you may want to raise the value of the vm subsystem attribute ubc_minpercent to ensure that enough memory is available for the UBC.

To ensure that the value of the ubc_minpercent is appropriate, use the vmstat command to examine the page-out rate. See Section 6.3.1 for information.

Recommended Values

The default value of the ubc_minpercent is 10 percent.

If the values of the vm subsystem attributes ubc_maxpercent and ubc_minpercent are close together, you may degrade I/O performance.

See Section 3.6 for information about modifying kernel subsystem attributes.

9.2.7 Improving Large File Caching Performance

If a large file completely fills the UBC, it may take all of the pages on the free page list, which may cause the system to page excessively. The vm subsystem attribute vm_ubcseqpercent specifies the maximum amount of memory allocated to the UBC that can be used to cache a single file.

The vm subsystem attribute vm_ubcseqstartpercent specifies the size of the UBC as a percentage of physical memory, at which time the virtual memory subsystem starts stealing the UBC LRU pages for a file to satisfy the demand for pages.

Performance Benefit and Tradeoff

Increasing the value of the vm_ubcseqpercent attribute will improve the I/O performance of a large single file, but will decrease the memory available for small files.

You can modify the vm_ubcseqpercent and vm_ubcseqstartpercent attributes without rebooting the system.

When to Tune

You may want to increase the value of the vm_ubcseqpercent attribute if you reuse large files.

Recommended Values

The default value of the vm_ubcseqpercent attribute is 10 percent of memory allocated to the UBC.

To force the system to reuse the pages in the UBC instead of taking pages from the free list, perform the following tasks:

- Make the maximum size of the UBC greater than the size of the UBC as a percentage of memory. That is, the value of the vm subsystem attribute ubc_maxpercent (the default is 100 percent) must be greater than the value of the vm_ubcseqstartpercent attribute (the default is 50 percent).
- Make the value of the vm_ubcseqpercent attribute, which specifies the size of a file as a percentage of the UBC, greater than a referenced file. The default value of the vm_ubcseqpercent attribute is 10 percent.

For example, using the default values, the UBC would have to be larger than 50 percent of all memory and a file would have to be larger than 10 percent of the UBC (that is, the file size would have to be at least 5 percent of all memory) in order for the system to reuse the pages in the UBC.

On large-memory systems that are doing a lot of file system operations, you may want to decrease the value of the vm_ubcseqstartpercent attribute to 30 percent. Do not specify a lower value unless you decrease the size of the UBC. In this case, do not change the value of the vm_ubcseqpercent attribute.

See Section 3.6 for information about modifying kernel subsystem attributes.

9.2.8 Disabling File Read Access Time Flushing

When a read system call is made to a file system's files, the default behavior is for the file system to update both the in-memory file access time and the on-disk stat structure, which contains most of the file information that is returned by the stat system call.

Performance Benefit and Tradeoff

You can improve file system performance for systems that perform mainly read operations (such as proxy servers) by specifying, at mount time, that the file system update only the in-memory file access time when a read system call is made to a file. The file system will update the on-disk stat structure only if the file is modified.

Updating only the in-memory file access time for reads can improve proxy server response time by decreasing the number of disk I/O operations. However, this behavior jeopardizes the integrity of read access time updates and violates POSIX standards. Do not use this functionality if it will affect

utilities that use read access times to perform tasks, such as migrating files to different devices.

When to Perform this Task

You may want to disable file read access time flushing if your system performs mainly read operations.

Recommended Procedure

To disable file read access time flushing, use the `mount` command with the `noatimes` option.

See `read`(2) and `mount`(8) for more information.

9.2.9 Caching Only File System Metadata with Prestoserve

Prestoserve can improve the overall run-time performance for systems that perform large numbers of synchronous writes. The `prmetaonly` attribute controls whether Prestoserve caches only UFS and AdvFS file system metadata, instead of both metadata and synchronous write data (the default).

Performance Benefit and Tradeoff

Caching only metadata may improve the performance of applications that access many small files or applications that access a large amount of file-system metadata but do not reread recently written data.

When to Tune

Cache only file system metadata if your applications access many small files or access a large amount of file-system metadata but do not reread recently written data.

Recommended Values

Set the value of the `prmetaonly` attribute to 1 (enabled) to cache only file system metadata.

See Section 3.6 for information about modifying kernel subsystem attributes.

9.3 Managing Advanced File System Performance

The Advanced File System (AdvFS) provides file system features beyond those of a traditional UFS file system. Unlike the rigid UFS model in which the file system directory hierarchy (tree) is bound tightly to the physical storage, AdvFS consists of two distinct layers: the directory hierarchy layer and the physical storage layer.

The following sections describe:

- AdvFS features (Section 9.3.1)
- AdvFS I/O queues (Section 9.3.2)
- AdvFS access structures (Section 9.3.3)
- AdvFS guidelines for high-performance configurations (Section 9.3.4)
- Monitoring AdvFS performance (Section 9.3.5)
- Tuning AdvFS (Section 9.3.6)
- Improving AdvFS performance (Section 9.3.7)

See the *AdvFS Administration* manual for detailed information about setting up and managing AdvFS.

9.3.1 AdvFS Features

The AdvFS decoupled file system structure enables you to manage the physical storage layer apart from the directory hierarchy layer. You can put multiple volumes (disks, LSM volumes, or RAID storage sets) in a file domain and distribute the filesets and files across the volumes. A file's blocks usually reside together on the same volume, unless the file is striped or the volume is full. Each new file is placed on the successive volume by using round-robin scheduling.

AdvFS enables you to move files between a defined group of disk volumes without changing file pathnames. Because the pathnames remain the same, the action is completely transparent to users.

The AdvFS Utilities product, which is licensed separately from the operating system, extends the capabilities of the AdvFS file system.

AdvFS provides the following basic features that do not require a license:

- High-performance file system

  AdvFS uses an extent-based file allocation scheme that consolidates data transfers, which increases sequential bandwidth and improves performance for large data transfers. AdvFS performs large reads from disk when it anticipates a need for sequential data. AdvFS also performs large writes by combining adjacent data into a single data transfer.

- Fast file system recovery

  Rebooting after a system interruption is extremely fast, because AdvFS uses write-ahead logging, instead of the `fsck` utility, as a way to check for and repair file system inconsistencies. The recovery speed depends on the number of uncommitted records in the log, not the amount of data in the fileset; therefore, reboots are quick and predictable.

- Direct I/O support

AdvFS allows you to enable direct I/O functionality on the files in a fileset or on a specific file. If direct I/O is enabled, file data is synchronously read or written without copying the data into the AdvFS buffer cache. Direct I/O can significantly improve disk I/O throughput for applications that read or write data only once or do not frequently write to previously written pages. See Section 9.3.4.7 for more information.

- Smooth sync

  Smooth sync functionality improves AdvFS asynchronous I/O performance by preventing I/O spikes caused by the `update` daemon, increasing the chance of a buffer cache hit, and improving the consolidation of I/O requests. See Section 9.3.6.5 for more information.

- Online file domain defragmentation capability

  Defragmenting disk data can improve performance by making data more contiguous. AdvFS enables you to perform this task without interrupting data availability.

- Disk quotas

  AdvFS enables you to track and control the amount of disk storage that each user, group, and fileset consumes.

The optional AdvFS utilities product, which requires a license, provides the following features:

- Disk spanning

  A file or fileset can span disks within a multi-volume file domain.

- Online file system resizing

  You can dynamically change the size of a file system by adding or removing disks. AdvFS enables you to perform this task without disrupting users or applications.

- Ability to recover deleted files

  Users can retrieve their own unintentionally deleted files from predefined trashcan directories, without assistance from system administrators.

- I/O load balancing across disks

  You can distribute the percentage of used space evenly between volumes in a multi-volume domain.

- Online file migration across disks

  You can move specific files to different volumes to eliminate bottlenecks caused by heavily used files.

- Online backup

  You can back up file system contents with limited interruption to users.

- Clone filesets

  AdvFS enables you to clone a fileset, which produces a read-only snapshot of fileset data structures. Cloning can increase the availability of data by preserving the state of the AdvFS data at a particular time and protecting against accidental file deletion or corruption.

- File-level striping

  File-level striping may improve I/O bandwidth (transfer rates) by distributing file data across multiple disk volumes.

- Graphical user interface

  The AdvFS GUI simplifies disk and file system administration, provides status, and alerts you to potential problems.

See the *AdvFS Administration* manual for detailed information about AdvFS features.

9.3.2 AdvFS I/O Queues

The AdvFS buffer cache is part of the UBC, and acts as a layer between the operating system and disk by storing recently accessed AdvFS file system data. Performance is improved if the cached data is later reused (a buffer cache hit) and a disk operation is avoided.

At boot time, the kernel determines the amount of physical memory that is available for AdvFS buffer cache headers, and allocates a buffer cache header for each possible page. The size of an AdvFS page is 8 KB.

The number of AdvFS buffer cache headers depends on the number of 8-KB pages that can be obtained from the amount of memory specified by the `advfs` subsystem attribute `AdvfsCacheMaxPercent`. The default value is 7 percent of physical memory. See Section 6.1.2.3 for more information about how the system allocates memory to the AdvFS buffer cache.

For each AdvFS volume, I/O requests are sent to one of the following queues, which feed I/O requests to the device queue:

- Blocking queue

  The blocking queue caches synchronous I/O requests. A synchronous I/O request is a read operation or a write that must be flushed to disk before it is considered complete and the application can continue. This ensures data reliability because the data has been written to disk and is not stored only in memory. Therefore, I/O requests on the blocking queue cannot be asynchronously removed, because the I/O must complete.

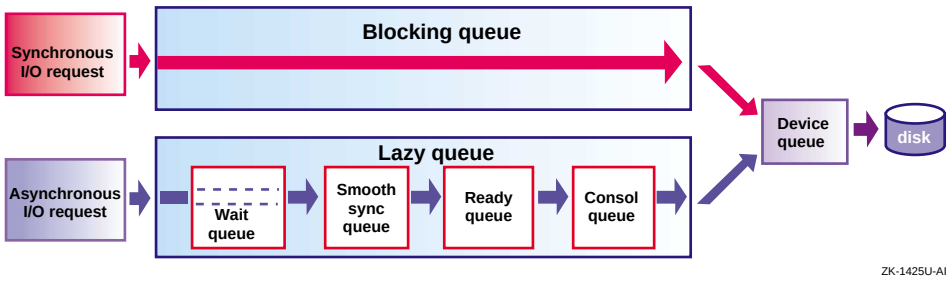
- Lazy queue

  The lazy queue caches asynchronous I/O requests. Asynchronous I/O requests are cached in the lazy queue and periodically flushed to disk

in portions that are large enough to allow the disk drivers to optimize the order of the write.

Figure 9–1 shows the movement of synchronous and asynchronous I/O requests through the AdvFS I/O queues.

Figure 9–1: AdvFS I/O Queues



When an asynchronous I/O request enters the lazy queue, it is assigned a time stamp. The lazy queue is a pipeline that contains a sequence of queues through which an I/O request passes: the wait queue (if applicable), the smooth sync queue, the ready queue, and the consol (consolidation) queue. An AdvFS buffer cache hit can occur while an I/O request is in any part of the lazy queue.

Detailed descriptions of the AdvFS 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 be moved from the wait queue to the smooth sync queue.

- Smooth sync queue—The smooth sync queue improves AdvFS asynchronous I/O performance by preventing I/O spikes caused by the `update` daemon, increasing the chance of an AdvFS buffer cache hit, and improving the consolidation of I/O requests.

  When smooth sync is not enabled, the `update` daemon flushes data from memory to disk every 30 seconds, regardless of how long a buffer has been cached. However, with smooth sync enabled (the default behavior), asynchronous I/O requests remain in the smooth sync queue for the amount of time specified by the value of the `vfs` attribute

smoothsync_age (the default is 30 seconds). After this time, the buffer moves to the ready queue. The movement of buffers from the smooth sync queue to the ready queue occurs continuously, based on the age of the buffer, and reduces the need to flush large numbers of requests every 30 seconds. See Section 9.3.6.5 for information about tuning the smooth sync queue.

- Ready queue—Asynchronous I/O requests that are not waiting for an AdvFS transaction log write to complete enter the ready queue, where they are sorted and held until the size of the ready queue reaches the value specified by the AdvfsReadyQLim attribute, or until the update daemon flushes the data. The default value of the AdvfsReadyQLim attribute is 16,384 512-byte blocks (8 MB).

  You can modify the size of the ready queue for all AdvFS volumes by changing the value of the AdvfsReadyQLim attribute. Alternatively, you can modify the ready queue limit for a specific AdvFS volume by using the chvol -t command. See Section 9.3.6.4 for information about tuning the ready queue.

- Consol queue—I/O requests are moved from the ready queue to the consol queue, which feeds the device queue. The consol queue serves as a holding area that enables the interleaving of I/O requests as they move from the blocking and the consol queues to the device queue, and also prevents flooding the device queue with requests.

Both the consol queue and the blocking queue feed the device queue, where logically contiguous I/O requests are consolidated into larger I/Os before they are sent to the device driver. The size of the device queue affects the amount of time it takes to complete a synchronous (blocking) I/O operation. AdvFS issues several types of blocking I/O operations, including AdvFS metadata and log data operations.

The AdvfsMaxDevQLen attribute limits the total number of I/O requests on the AdvFS device queue. The default value is 24 requests. When the number of requests exceeds this value, only synchronous requests from the blocking queue are accepted onto the device queue.

Although the default value of the AdvfsMaxDevQLen attribute is appropriate for most configurations, you may need to modify this value. However, increase the default value only if devices are not being kept busy. Make sure that increasing the size of the device queue does not cause a decrease in response time. See Section 9.3.6.6 for more information about tuning the AdvFS device queue.

Use the advfsstat command to show the AdvFS queue statistics. See Section 9.3.5.1 for information.

9.3.3 AdvFS Access Structures

AdvFS access structures are in-memory data structures that AdvFS uses to cache low-level information about files that are currently open and files that were opened but are now closed. Caching open file information can enhance AdvFS performance if the open files are later reused. If your users or applications open and then reuse many files, you may be able to improve AdvFS performance by modifying how the system allocates AdvFS access structures.

There are three attributes that control the allocation of AdvFS access structures:

- The `AdvfsAccessMaxPercent` attribute controls the maximum percentage of pageable memory that can be allocated for AdvFS access structures.
- At boot time, the system reserves for AdvFS access structures a portion of the physical memory that is not wired. The memory reserved is either twice the value of the `AdvfsMinFreeAccess` attribute or the value of the `AdvfsAccessMaxPercent` attribute, whichever is smaller. These access structures are then placed on the access structure free list.

  As AdvFS files are opened, access structures are taken from the free list. If the number of access structures on the free list falls below the value of the `AdvfsMinFreeAccess` attribute, AdvFS allocates additional access structures and places them on the free list, until the number of access structures on the free list is twice the value of the `AdvfsMinFreeAccess` attribute or the value of the `AdvfsAccessMaxPercent` attribute, whichever is smaller.
- At any one time, the access structure free list contains only a portion of the access structures that the system has allocated. The `AdvfsMaxFreeAccessPercent` attribute specifies the maximum percentage of the total allocated access structures that can be on the free list at one time. Access structures are deallocated from the free list, and memory is returned to the pool that is reserved for access structures when the following occurs:
  - The number of access structures on the free list exceeds the value of the `AdvfsMaxFreeAccessPercent` attribute (as a percentage of the total allocated access structures). For example, this condition is satisfied if the value of the `AdvfsMaxFreeAccessPercent` attribute is 80 percent, there are 100 allocated access structures, and the number of access structures on the free list is more then 80.
  - The number of access structures on the free list is more than twice the value of the `AdvfsMinFreeAccess` attribute.

You may be able to improve AdvFS performance by modifying the previous attributes and allocating more memory for AdvFS access structures. However, this will reduce the amount of memory available to processes and may cause excessive paging and swapping. See Section 9.3.6.3 for information.

If you do not use AdvFS or if your workload does not frequently write to previously-written pages, do not allocate a large amount of memory for access structures. If you have a large-memory system, you may want to decrease the amount of memory reserved for AdvFS access structures. See Section 6.4.5 for information.

9.3.4 AdvFS Configuration Guidelines

You will obtain the best performance if you carefully plan your AdvFS configuration. Table 9–2 lists AdvFS configuration guidelines and performance benefits as well as tradeoffs. See the *AdvFS Administration* manual for detailed information about AdvFS configuration.

Table 9–2: AdvFS Configuration Guidelines

Guideline	Performance Benefit	Tradeoff
Use a few file domains instead of a single large domain (Section 9.3.4.1)	Facilitates administration	None
Use a multi-volume file domains, instead of single-volume domains (Section 9.3.4.1)	Improves throughput	Multi-volumes increase the chance of domain failure
Configure one fileset for each domain (Section 9.3.4.2)	Facilitates administration	None
Keep filesets less than 50 GB in size (Section 9.3.4.2)	Facilitates administration	None
Distribute the I/O load over multiple disks (Section 9.3.4.3)	Improves throughput	Requires multiple disks
Place the transaction log on fast or uncongested volume (Section 9.3.4.4)	Prevents the log from becoming a bottleneck	None
Log only file structures (Section 9.3.4.4)	Maintains high performance	Increases the possibility of inconsistent data after a crash
Force all AdvFS file writes to be synchronous (Section 9.3.4.5)	Ensures that data is successfully written to disk	May degrade file system performance

Table 9–2: AdvFS Configuration Guidelines (cont.)

Guideline	Performance Benefit	Tradeoff
Prevent partial writes (Section 9.3.4.6)	Ensures that system crashes do not cause partial disk writes	May degrade asynchronous write performance
Enable direct I/O (Section 9.3.4.7)	Improves disk I/O throughput for database applications that read or write data only once	Degrades I/O performance for applications that repeatedly access the same data
Use AdvFS for the root file system (Section 9.3.4.8)	Provides fast startup after a crash	None
Stripe files across different disks and, if possible, different buses (Section 9.3.4.9)	Improves sequential read and write performance	Increases chance of domain failure
Use quotas (Section 9.3.4.10)	Tracks and controls the amount of disk storage that each user, group, or fileset consumes	None
Consolidate I/O transfers (Section 9.3.4.11)	Improves AdvFS performance	None
Allocate sufficient swap space (Section 2.3.2.3)	Facilitates the use of the `verify` command	Requires additional disk space

The following sections describe these AdvFS configuration guidelines in detail.

9.3.4.1 Configuring File Domains

To facilitate AdvFS administration and improve performance, configure a few file domains with multiple volumes instead of many file domains or a single large file domain. Using a few file domains with multiple volumes provides better control over physical resources, improves a fileset's total throughput, and decreases the administration time.

Each file domain uses a transaction log on one of the volumes. If you configure only a single large multi-volume file domain, the log may become a bottleneck. In contrast, if you configure many file domains, you spread the overhead associated with managing the logs for the file domains.

Multi-volume file domains improve performance because AdvFS generates parallel streams of output using multiple device consolidation queues. A file domain with three volumes on different disks is more efficient than a file domain consisting of a single disk because the latter has only one I/O path.

However, a single volume failure within a file domain will render the entire domain inaccessible, so the more volumes that you have in a file domain, the greater the risk that the domain will fail. To reduce the risk of file domain failure, limit the number of volumes in a file domain to eight or mirror the file domain with LSM or hardware RAID.

In addition, follow these guidelines for configuring file domains:

- For the best efficiency, spread a file domain across several of the same type of disks with the same speed.
- Use an entire disk in a file domain. For example, do not use partition `a` in one file domain and partition `b` in another file domain.
- Use a single disk partition to add a disk to a file domain (for example, partition `c`), instead of using multiple partitions.
- Make sure that busy files are not located on the same volume. Use the `migrate` command to move files across volumes.
- If you are using LSM, use multiple, small LSM volumes in a file domain, instead of a single, large concatenated or striped volume. This enables AdvFS to balance I/O across volumes.

9.3.4.2 Configuring Filesets for High Performance

Configuring many filesets in a file domain can adversely affect performance and AdvFS administration. If possible, configure only one fileset for each file domain.

In addition, the recommended maximum size of a fileset is 50 GB. Once a fileset reaches 30 GB, consider creating another file domain and fileset. You may want to establish a monitoring routine that alerts you to a large fileset size.

Use the `showfsets` command to display the number of filesets in a domain and the size of a fileset. See `showfsets`(8) for more information.

9.3.4.3 Distribute the AdvFS I/O Load

Distribute the AdvFS I/O load over multiple disks to improve throughput. Use multiple file domains and spread filesets across the domains.

The number of filesets depends on your storage needs. Each fileset can be managed and backed up independently, and can be assigned quotas. Be sure that heavily used filesets are located on different file domains, so that a single transaction log does not become a bottleneck.

See Section 8.1 for more information about distributing the disk I/O load.

9.3.4.4 Improving the Transaction Log Performance

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. The AdvFS file domain transaction log may become a bottleneck if the log resides on a congested disk or bus, or if the file domain contains many filesets.

To prevent the log from becoming a bottleneck, put the log on a fast, uncongested volume. You may want to put the log on a disk that contains only the log. See Section 9.3.7.3 for information on moving an existing transaction log.

To make the transaction log highly available, use LSM or hardware RAID to mirror the log.

You can also divide a large multi-volume file domain into smaller file domains to distribute transaction log I/O.

By default, AdvFS logs only file structures. However, you can also log file data to ensure that a file is internally consistent if a crash occurs. However, data logging can degrade performance. See Section 9.3.4.6 for information about atomic write data logging.

9.3.4.5 Forcing Synchronous Writes

By default, asynchronous write requests are cached in the AdvFS buffer cache, and the `write` system call then returns a success value. The data is written to disk at a later time (asynchronously).

Use the `chfile -l on` command to force all write requests to a specified AdvFS file to be synchronous. If you enable forced synchronous writes on a file, data must be successfully written to disk before the `write` system call will return a success value. This behavior is similar to the behavior associated with a file that has been opened with the `O_SYNC` option; however, forcing synchronous writes persists across `open` calls.

Forcing all writes to a file to be synchronous ensures that the write has completed when the `write` system call returns a success value. However, it may degrade write performance.

A file cannot have both forced synchronous writes enabled and atomic write data logging enabled. See Section 9.3.4.6 for more information.

Use the `chfile` command to determine whether forced synchronous writes or atomic write data logging is enabled. Use the `chfile -l off` command to disable forced synchronous writes (the default).

9.3.4.6 Preventing Partial Data Writes

AdvFS writes data to disk in 8-KB chunks. By default, and in accordance with POSIX standards, 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 chunks of data, only a portion (anywhere from 0 to 16 KB) of the total write may have succeeded. This can result in partial data writes and inconsistent data.

To prevent partial writes if a system crash occurs, use the `chfile -L on` command to enable atomic write data logging for a specified file.

By default, each file domain has a transaction log file that tracks fileset activity and ensures that AdvFS can maintain a consistent view of the file system metadata if a crash occurs. If you enable atomic write data logging on a file, data from a write call will be written to the transaction log file before it is written to disk. If a system crash occurs during or immediately after the write call, upon recovery, the data in the log file can be used to reconstruct the write. This guarantees that each 8-KB chunk of a write either is completely written to disk or is not written to disk.

For example, if atomic write data logging is enabled and a crash occurs during a write that consists of two 8-KB chunks of data, the write can have three possible states: none of the data is written, 8 KB of the data is written, or 16 KB of data is written.

Atomic write data logging may degrade AdvFS write performance because of the extra write to the transaction log file. In addition, a file that has 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 9.3.4.7).

A file cannot have both forced synchronous writes enabled (see Section 9.3.4.5) and atomic write data logging enabled. 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.

Use the `chfile` command to determine if forced synchronous writes or atomic write data logging is enabled. Use the `chfile -L off` command to disable atomic write data logging (the default).

To enable atomic write data logging on AdvFS files that are NFS mounted, the NFS property list daemon, `proplistd`, must be running on the NFS client and the fileset must be mounted on the client by using the `mount` command's `proplist` option.

If atomic write data logging is enabled and you are writing to a file that has been NFS mounted, the offset into the file must be on an 8-KB page boundary, because NFS performs I/O on 8-KB page boundaries.

You can also activate and deactivate atomic data logging by using the `fcntl` system call. In addition, both the `chfile` command and `fcntl` can be used on an NFS client to activate or deactivate this feature on a file that resides on the NFS server.

9.3.4.7 Enabling Direct I/O

You can use direct I/O to read and write data from a file without copying the data into the AdvFS buffer cache. If you enable direct I/O, read and write requests are executed to and from disk through direct memory access, bypassing the AdvFS buffer cache.

Direct I/O can significantly improve disk I/O throughput for database applications that read or write data only once (or for applications that do not frequently write to previously written pages). However, direct I/O can degrade disk I/O performance for applications that access data multiple times, because data is not cached. As soon as you specify direct I/O, any data already in the buffer cache is automatically flushed to disk.

If you enable direct I/O, by default, reads and writes to a file will be done synchronously. However, 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. See the *Programmer's Guide* for more information.

Although direct I/O will handle I/O requests of any byte size, the best performance will occur when the requested byte size is aligned on file page boundaries and is evenly divisible into 8-KB pages. Direct transfer from the user buffer to the disk is optimized in this case.

To enable direct I/O for a specific file, use the `open` system call and set the `O_DIRECTIO` file access flag. Once a file is opened for direct I/O, this mode is in effect until all users close the file.

Note that 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.

See `fcntl`(2), `open`(2), *AdvFS Administration*, and the *Programmer's Guide* for more information.

9.3.4.8 Configuring an AdvFS root File system

There are several advantages to configuring an AdvFS root file system:

- Quick restart after a crash, because you do not run the `fsck` utility after a crash.
- One set of tools to manage all local file systems. All features of AdvFS except `addvol` and `rmvol` are available to manage the root file system.
- Use AdvFS with LSM to mirror the root file system. This allows your root file system to remain viable even if there is a disk failure.

You can configure an AdvFS root file system during the initial base-system installation, or you can convert your existing root file system after installation. See the *AdvFS Administration* manual for more information.

9.3.4.9 Striping Files

You may be able to use the AdvFS `stripe` utility to improve the sequential read and write performance of an individual file by spreading file data evenly across different disks in a file domain. For the maximum performance benefit, stripe files across disks on different I/O buses.

Striping files, instead of striping entire disks with RAID 0, is useful if an application continually accesses only a few specific files. Do not stripe both a file and the disk on which it resides. For information about striping entire disks, see Chapter 8.

The `stripe` utility distributes a zero-length file (a file with no data written to it yet) evenly across a specified number of volumes. As data is appended to the file, the data is spread across the volumes. The size of each data segment (also called the stripe or chunk size) is fixed at 64 KB (65,536 bytes). AdvFS alternates the placement of the segments on the disks in a sequential pattern. For example, the first 64 KB of the file is written to the first volume, the second 64 KB is written to the next volume, and so on.

If an application's I/O transfer read or write size is more than 64 KB, striping files may improve application performance by enabling parallel I/O operations on multiple controllers or volumes, because AdvFS file striping uses a fixed 64 KB stripe width.

Note

Distributing data across multiple volumes decreases data availability, because one volume failure makes the entire file domain unavailable. To make striped files highly available, you can use RAID 1 to mirror the disks across which the file is striped. For information about mirroring, see Chapter 8.

See stripe(8) for more information.

9.3.4.10 Using AdvFS Quotas

AdvFS quotas allow you to track and control the amount of physical storage that a user, group, or fileset consumes. In addition, AdvFS quota information is always maintained, but quota enforcement can be activated and deactivated.

You can set quota values on the amount of disk storage and on the number of files. Quotas that apply to users and groups are similar to UFS quotas. You can set a separate quota for each user or each group of users for each fileset.

In addition, you can restrict the space that a fileset itself can use. Fileset quotas are useful when a file domain contains multiple filesets. Without fileset quotas, any fileset can consume all of the disk space in the file domain.

All quotas can have two types of limits: hard and soft. A hard limit cannot be exceeded; space cannot be allocated and files cannot be created. A soft limit permits a period of time during which the limit can be exceeded as long as the hard limit has not been exceeded.

For information about AdvFS quotas, see the *AdvFS Administration* manual.

9.3.4.11 Consolidating I/O Transfers

By default, AdvFS consolidates a number of I/O transfers into a single, large I/O transfer, which can improve AdvFS performance. To enable the consolidation of I/O transfers, use the chvol command with the –c on option.

It is recommended that you not disable the consolidation of I/O transfers. See chvol(8) for more information.

9.3.5 Gathering AdvFS Information

Table 9–3 describes the tools you can use to obtain information about AdvFS.

Table 9–3: AdvFS Monitoring Tools

Name	Use	Description
advfsstat	Displays AdvFS performance statistics (Section 9.3.5.1)	Allows you to obtain extensive AdvFS performance information, including buffer cache, fileset, volume, and bitfile metadata table (BMT) statistics, for a specific interval of time.

Table 9–3: AdvFS Monitoring Tools (cont.)

Name	Use	Description
advscan	Identifies disks in a file domain (Section 9.3.5.2)	Locates pieces of AdvFS file domains on disk partitions and in LSM disk groups.
showfdmn	Displays detailed information about AdvFS file domains and volumes (Section 9.3.5.3)	Allows you to determine if file data is evenly distributed across AdvFS volumes. The showfdmn utility displays 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. For multivolume domains, the utility also displays the total volume size, the total number of free blocks, and the total percentage of volume space currently allocated.
showfile	Displays information about files in an AdvFS fileset (Section 9.3.5.4)	Displays detailed information about files (and directories) in an AdvFS fileset. The showfile command allows you to check a file's fragmentation. A low performance percentage (less than 80 percent) indicates that the file is fragmented on the disk. The showfile command also displays the extent map of each file. An extent is a contiguous area of disk space that AdvFS allocates to a file. Simple files have one extent map; striped files have an extent map for every stripe segment. The extent map shows whether the entire file or only a portion of the file is fragmented.
showfsets	Displays AdvFS fileset information for a file domain (Section 9.3.5.5)	Displays 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. The showfsets command also displays block and file quota limits for a file domain or for a specific fileset in the domain.
quota	Displays disk usage and quota limits	Displays the block usage, number of files, and quotas for a user or group. You can choose to display quota information for users or groups, for all filesets with usage over quota, or for all mounted filesets regardless of whether quotas are activated. See quota(1) for more information.

Table 9–3: AdvFS Monitoring Tools (cont.)

Name	Use	Description
vdf	Clarifies the relationship between file domain and fileset disk usage	Reformats output from the showfdmn, showfsets, shfragbf, and df commands to display information about the disk usage of AdvFS file domains and filesets. It clarifies the relationship between a domain's disk usage and its fileset's disk usage. See vdf(8) for more information.
vbmtpg	Displays a formatted page of the BMT (Section 9.3.5.6)	The vbmtpg utility displays a complete, formatted page of the BMT for a mounted or unmounted AdvFS domain. This utility is useful for debugging when there has been some seemingly random file corruption.

The following sections describe some of these commands in detail.

9.3.5.1 Monitoring AdvFS Performance Statistics by Using the advfsstat Command

The advfsstat command displays various AdvFS performance statistics and monitors the performance of AdvFS domains and filesets. Use this command to obtain detailed information, especially if the iostat command output indicates a disk bottleneck (see Section 8.2).

The advfsstat command displays detailed information about a file domain, including information about the AdvFS buffer cache, fileset vnode operations, locks, the namei cache, and volume I/O performance. The command reports information in units of one disk block (512 bytes). By default, the command displays one sample. You can use the -i option to output information at specific time intervals.

The following example of the advfsstat -v 2 command shows the current I/O queue statistics for the specified file domain:

```
# /usr/sbin/advfsstat -v 2 test_domain
vol1    rd  wr  rg  arg  wg  awg  blk  wlz  sms rlz  con  dev
        54   0  48  128   0    0    0    1    0   0    0   65
```

The previous example shows the following fields:

- Read and write requests—Compare the number of read requests (rd) to the number of write requests (wr). Read requests are blocked until the read completes, but write requests will not block the calling thread, which increases the throughput of multiple threads.
- Consolidated reads and writes—You may be able to improve performance by consolidating reads and writes. The consolidated read values (rg and arg) and write values (wg and awg) 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.

- I/O queue values—The `blk`, `wlz`, `sms`, `rlz`, `con`, and `dev` fields can indicate potential performance issues. The `sms` value specifies the number of requests on the smooth sync queue. The `con` value specifies the number of entries on the consolidate queue. These entries are ready to be consolidated and moved to the device queue. The device queue value (`dev`) shows the number of I/O requests that have been issued to the device controller. The system must wait for these requests to complete.

  If the number of I/O requests on the device queue increases continually and you experience poor performance, applications may be I/O bound on this device. You may be able to eliminate the problem by adding more disks to the domain or by striping with LSM or hardware RAID.

You can monitor the type of requests that applications are issuing by using the `advfsstat` command's `-f` option to display fileset vnode operations. You can display the number of file creates, reads, and writes and other operations for a specified domain or fileset. For example:

```
# /usr/sbin/advfsstat -i 3 -f 2 scratch_domain fset1
  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.

Note that it is difficult to link performance problems to some statistics such as buffer cache statistics. In addition, lock performance that is related to lock statistics cannot be tuned.

9.3.5.2 Identifying Disks in an AdvFS File Domain by Using the advscan Command

The `advscan` command locates pieces of AdvFS domains on disk partitions and in LSM disk groups. Use the `advscan` command when you have moved disks to a new system, have moved disks in a way that has changed device numbers, or have lost track of where the domains are.

You can specify a list of volumes or disk groups with the advscan command to search all partitions and volumes. The command determines which partitions on a disk are part of an AdvFS file domain.

You can also use the advscan command for repair purposes if you deleted the /etc/fdmns directory, deleted a directory domain under /etc/fdmns, or deleted some links from a domain directory under /etc/fdmns.

Use the advscan command to rebuild all or part of your /etc/fdmns directory, or you can manually rebuild it by supplying the names of the partitions in a domain.

The following example scans two disks for AdvFS partitions:

```
# /usr/advfs/advscan dsk0 dsk5
Scanning disks  dsk0 dsk5
Found domains:
usr_domain
          Domain Id       2e09be37.0002eb40
          Created         Thu Jun 26 09:54:15 1998
          Domain volumes          2
          /etc/fdmns links        2
          Actual partitions found:
                                  dsk0c
                                  dsk5c
```

For the following example, the dsk6 file domains were removed from /etc/fdmns. The advscan command scans device dsk6 and re-creates the missing domains.

```
# /usr/advfs/advscan -r dsk6
Scanning disks  dsk6
Found domains: *unknown*
          Domain Id       2f2421ba.0008c1c0
          Created         Mon Jan 20 13:38:02 1998
          Domain volumes          1
          /etc/fdmns links        0
          Actual partitions found:
                                  dsk6a*
*unknown*
         Domain Id       2f535f8c.000b6860
         Created         Tue Feb 25 09:38:20 1998
         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.

9.3.5.3 Checking AdvFS File Domains by Using the showfdmn Command

The showfdmn command displays the attributes of an AdvFS file domain and detailed information about each volume in the file domain.

The following example of the showfdmn command displays domain information for the root_domain file domain:

```
 % /sbin/showfdmn root_domain
               Id              Date Created  LogPgs  Version  Domain Name
34f0ce64.0004f2e0  Wed Mar 17 15:19:48 1999     512        4  root_domain

  Vol   512-Blks        Free  % Used  Cmode  Rblks  Wblks  Vol Name
   1L     262144       94896     64%     on    256    256  /dev/disk/dsk0a
```

See showfdmn(8) for more information about the output of the command.

9.3.5.4 Displaying AdvFS File Information by Using the showfile Command

The showfile command displays the full storage allocation map (extent map) for one or more files in an AdvFS fileset. An extent is a contiguous area of disk space that AdvFS allocates to a file.

The following example of the showfile command displays the AdvFS characteristics for all of the files in the current working directory:

```
# /usr/sbin/showfile *

        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
```

The I/O column specifies whether write operations are forced to be synchronous. See Section 9.3.4.5 for information.

The following example of the showfile command shows the characteristics and extent information for the tutorial file, which is a simple file:

```
# /usr/sbin/showfile -x tutorial

       Id  Vol  PgSz  Pages  XtntType  Segs  SegSz    I/O  Perf    File
4198.800d    2    16     27    simple    **     **  async   66% tutorial
```

```
extentMap: 1
     pageOff    pageCnt    vol    volBlock    blockCnt
           0          5      2      781552          80
           5         12      2      785776         192
          17         10      2      786800         160
  extentCnt: 3
```

The `Perf` entry shows the efficiency of the file-extent allocation, expressed as a percentage of the optimal extent layout. A high value, such as 100 percent, indicates that the AdvFS I/O subsystem is highly efficient. A low value indicates that files may be fragmented.

See `showfile`(8) for more information about the command output.

9.3.5.5 Displaying the AdvFS Filesets in a File Domain by Using the showfsets Command

The `showfsets` command displays the AdvFS filesets (or clone filesets) and their characteristics in a specified domain.

The following is an example of the `showfsets` command shows that the `dmn1` file domain has one fileset and one clone fileset:

```
# /sbin/showfsets dmn1
mnt
  Id           : 2c73e2f9.000f143a.1.8001
  Clone is     : mnt_clone
  Files        :     7456,  SLim= 60000, HLim=80000
  Blocks  (1k) :   388698,  SLim= 6000,  HLim=8000
  Quota Status : user=on  group=on

mnt_clone
  Id           : 2c73e2f9.000f143a.2.8001
  Clone of     : mnt
  Revision     : 2
```

See `showfsets`(8) for information about the options and output of the command.

9.3.5.6 Monitoring the Bitmap Metadata Table

The AdvFS fileset data structure (metadata) is stored in a file called the bitfile metadata table (BMT). Each volume in a domain has a BMT that describes the file extents on the volume. If a domain has multiple volumes of the same size, files will be distributed evenly among the volumes.

The BMT is the equivalent of the UFS inode table. However, the UFS inode table is statically allocated, while the BMT expands as more files are added to the domain. Each time AdvFS needs additional metadata, the BMT grows by a fixed size (the default is 128 pages). As a volume becomes increasingly fragmented, the size by which the BMT grows may be described by several extents.

To monitor the BMT, use the `vbmtpg` command and examine the number of mcells (`freeMcellCnt`). The value of `freeMcellCnt` can range from 0 to 22. A volume with 1 free mcell has very little space in which to grow the BMT. See `vbmtpg`(8) for more information.

You can also invoke the `showfile` command and specify *mount_point*`/.tags/M-10` to examine the BMT extents on the first domain volume that contains the fileset mounted on the specified mount point. To examine the extents of the other volumes in the domain, specify `M-16`, `M-24`, and so on. If the extents at the end of the BMT are smaller than the extents at the beginning of the file, the BMT is becoming fragmented. See `showfile`(8) for more information.

9.3.6 Tuning AdvFS

After you configure AdvFS, as described in Section 9.3.4, you may be able to tune it to improve performance. To successfully improve performance, you must understand how your applications and user perform file system I/O, as described in Section 2.1.

Table 9–4 lists AdvFS tuning guidelines and performance benefits as well as tradeoffs. The guidelines described in Table 9–1 also apply to AdvFS configurations.

Table 9–4: AdvFS Tuning Guidelines

Guideline	Performance Benefit	Tradeoff
Decrease the size of the metadata buffer cache to 1 percent (Section 6.4.6)	Improves performance for systems that use only AdvFS	None
Increase the percentage of memory allocated for the AdvFS buffer cache (Section 9.3.6.1)	Improves AdvFS performance if data reuse is high	Consumes memory
Increase the size of the AdvFS buffer cache hash table (Section 9.3.6.2)	Speeds lookup operations and decreases CPU usage	Consumes memory
Increase the memory reserved for AdvFS access structures (Section 9.3.6.3)	Improves AdvFS performance for systems that open and reuse files	Consumes memory
Increase the amount of data cached in the ready queue (Section 9.3.6.4)	Improves AdvFS performance for systems that open and reuse files	May cause I/O spikes or increase the number of lost buffers if a crash occurs
Increase the smooth sync caching threshold for asynchronous I/O requests (Section 9.3.6.5)	Improves performance of AdvFS asynchronous I/O	Increases the chance that data may be lost if a system crash occurs

Table 9–4: AdvFS Tuning Guidelines (cont.)

Guideline	Performance Benefit	Tradeoff
Increase the maximum number of I/O requests on the device queue (Section 9.3.6.6)	Keeps devices busy	May degrade response time
Disable the flushing of dirty pages mapped with the `mmap` function during a `sync` call (Section 9.3.6.7)	May improve performance for applications that manage their own flushing	None

The following sections describe the AdvFS tuning guidelines in detail.

9.3.6.1 Increasing the Size of the AdvFS Buffer Cache

The `advfs` subsystem attribute `AdvfsCacheMaxPercent` specifies the maximum percentage of physical memory that can be used to cache AdvFS file data. Caching AdvFS data improves I/O performance only if the cached data is reused.

Performance Benefit and Tradeoff

If data reuse is high, you may be able to improve AdvFS performance by increasing the percentage of memory allocated to the AdvFS buffer cache. However, this will decrease the amount of memory available for processes.

You also may need to increase the number of AdvFS buffer cache hash chains to increase the size of the AdvFS buffer cache. See Section 9.3.6.2 for information.

You cannot modify the `AdvfsCacheMaxPercent` attribute without rebooting the system.

When to Tune

You may need to increase the size of the AdvFS buffer cache if data reuse is high and if pages are being rapidly recycled. Increasing the size of the buffer cache will enable pages to remain in the cache for a longer period of time. This increases the chance that a cache hit will occur.

Use the `advfsstat -b` command to determine if pages are being recycled too quickly. If the command output shows that the ratio of total hits (`hit`) to total counts (`cnt`), for both `pin` and `ref`, is less than 85 percent, pages are being rapidly recycled.

Recommended Values

The default value of the `AdvfsCacheMaxPercent` attribute is 7 percent of memory. The minimum value is 1 percent; the maximum value is 30 percent.

Increase the value of the AdvfsCacheMaxPercent attribute only by small increments to optimize file system performance without wasting memory. If you increase the value of the AdvfsCacheMaxPercent attribute and experience no performance benefit, return to the original value.

Use the vmstat command to check virtual memory statistics, as described in Section 6.3.1. Make sure that increasing the size of the AdvFS buffer cache does not cause excessive paging and swapping.

See Section 3.6 for information about modifying kernel subsystem attributes.

9.3.6.2 Increasing the Number of AdvFS Buffer Hash Chains

The buffer cache hash table for the AdvFS buffer cache is used to locate pages of AdvFS file data in memory. The table contains a number of hash chains, which contain elements that point to pages of file system data that have already been read into memory. When a read or write system call is done for a particular offset within an AdvFS file, the system sequentially searches the appropriate hash chain to determine if the file data is already in memory.

The value of the advfs subsystem attribute AdvfsCacheHashSize specifies the number of hash chains (entries) on the AdvFS buffer cache hash table.

Performance Benefit and Tradeoff

Increasing the number of hash chains on the buffer cache hash table will result in shorter hash chains. Short hash chains contain less elements to search, which increases search speeds and decreases CPU usage.

Increasing the size of the AdvFS buffer cache hash table will increase the amount of wired memory.

You cannot modify the AdvfsCacheHashSize attribute without rebooting the system.

When to Tune

If you have more than 4 GB of memory, you may want to increase the value of the AdvfsCacheHashSize attribute, which will increase the number of hash chains on the table.

To determine if your system performance may benefit from increasing the size of the buffer hash table, divide the number of AdvFS buffers by the current value of the AdvfsCacheHashSize attribute. Use the sysconfig -q advfs AdvfsCacheHashSize to determine the current value of the attribute. To obtain the number of AdvFS buffers, examine the AdvFS system initialization message that reports this value and the total amount of memory being used.

The result of the previous calculation will show the average number of buffers for each buffer hash table chain. A small number means fewer potential buffers that AdvFS must search. This assumes that buffers are evenly distributed across the AdvFS buffer cache hash table. If the average number of buffers for each chain is greater that 100, you may want to increase the size of the hash chain table.

Recommended Values

The default value of the `AdvfsCacheHashSize` attribute is either 8192 KB or 10 percent of the size of the AdvFS buffer cache (rounded up to the next power of 2), whichever is the smallest value. The minimum value is 1024 KB. The maximum value is either 65536 or the size of the AdvFS buffer cache, whichever is the smallest value. The `AdvfsCacheMaxPercent` attribute specifies the size of the AdvFS buffer cache (see Section 9.3.6.1).

You may want to double the default value of the `AdvfsCacheHashSize` attribute if the system is experiencing high CPU system time (see Section 6.3.1), or if a kernel profile shows high percentage of CPU usage in the `find_page` routine.

See Section 3.6 for information about modifying kernel subsystem attributes.

9.3.6.3 Increasing the Memory for Access Structures

AdvFS access structures are in-memory data structures that AdvFS uses to cache low-level information about files that are currently open and files that were opened but are now closed. Caching open file information can enhance AdvFS performance if the open files are later reused.

At boot time, the system reserves for AdvFS access structures a portion of physical memory that is not wired. Access structures are placed on the access structure free list, and are allocated and deallocated according to the kernel configuration and workload demands.

There are three attributes that control the allocation of AdvFS access structures:

- The `AdvfsAccessMaxPercent` attribute controls the maximum percentage of pageable memory that can be allocated for AdvFS access structures.
- At boot time, and when the number of access structures on the free list is less than the value of the `AdvfsMinFreeAccess` attribute, AdvFS allocates additional access structures, until the number of access structures on the free list is twice the value of the `AdvfsMinFreeAccess` attribute or the value of the `AdvfsAccessMaxPercent` attribute, whichever is smaller.

- The AdvfsMaxFreeAccessPercent attribute controls when access structures are deallocated from the free list. When the percentage of access structures on the free list is more than the value of the AdvfsMaxFreeAccessPercent attribute, and the number of access structures on the free list is more than twice the value of the AdvfsMinFreeAccess attribute, AdvFS deallocates access structures.

See Section 9.3.3 for information about access structures and attributes.

Performance Benefit and Tradeoff

Increasing the value of the AdvfsAccessMaxPercent attribute allows you to allocate more memory resources for access structures, which may improve AdvFS performance on systems that open and reuse many files. However, this increases memory consumption.

If you increase the value of the AdvfsMinFreeAccess attribute, you will retain more access structures on the free list and delay access structure deallocation, which may improve AdvFS performance for systems that open and reuse many files. However, this increases memory consumption.

If you increase the value of the AdvfsMaxFreeAccessPercent attribute, the system will retain access structures on the free list for a longer time, which may improve AdvFS performance for systems that open and reuse many files.

You can modify the AdvfsAccessMaxPercent, AdvfsMinFreeAccess, and AdvfsMaxFreeAccessPercent attributes without rebooting the system.

When to Tune

If your users or applications open and then reuse many AdvFS files (for example, if you have a proxy server), you may be able to improve AdvFS performance by increasing memory resources for access structures.

If you do not use AdvFS, if your workload does not frequently write to previously written pages, or if you have a large-memory system, you may want to decrease the memory allocated for access structures. See Section 6.4.5 for information.

Recommended Values

The default value of the AdvfsAccessMaxPercent attribute is 25 percent of pageable memory. The minimum value is 5 percent; the maximum value is 95 percent.

The default value of the AdvfsMinFreeAccess attribute is 128. The minimum value is 1; the maximum value is 100,000.

The default value of the AdvfsMaxFreeAccessPercent attribute is 80 percent. The minimum value is 5 percent; the maximum value is 95 percent.

See Section 3.6 for information about modifying kernel subsystem attributes.

9.3.6.4 Increasing Data Cached in the Ready Queue

AdvFS caches asynchronous I/O requests in the AdvFS buffer cache. If the cached data is later reused, pages can be retrieved from memory and a disk operation is avoided.

Asynchronous I/O requests are sorted in the ready queue and remain there until the size of the queue reaches the value specified by the `AdvfsReadyQLim` attribute or, if smooth sync is not enabled, until the `update` daemon flushes the data. See Section 9.3.2 for more information about AdvFS queues. See Section 9.3.6.5 for information about using smooth sync to control asynchronous I/O request caching.

Performance Benefit and Tradeoff

Increasing the size of the ready queue can improve AdvFS performance if data is reused by increasing the time that a buffer will stay on the I/O queue and not be flushed to disk.

You can modify the `AdvfsReadyQLim` attribute without rebooting the system.

When to Tune

If you have high data reuse (data is repeatedly read and written), you may want to increase the size of the ready queue. This can increase the number of AdvFS buffer cache hits. If you have low data reuse, it is recommended that you use the default value.

Recommended Values

You can modify the size of the ready queue for all AdvFS volumes by changing the value of the `AdvfsReadyQLim` attribute. The default value of the `AdvfsReadyQLim` attribute is 16,384 512-byte blocks (8 MB).

You can modify the size for a specific AdvFS volume by using the `chvol -t` command. See `chvol`(8) for more information.

If you change the size of the ready queue and performance does not improve, return to the original value.

See Section 3.6 for information about modifying kernel subsystem attributes.

9.3.6.5 Increasing the AdvFS Smooth Sync Cache Timeout Value

Smooth sync functionality improves AdvFS asynchronous I/O performance by preventing I/O spikes caused by the `update` daemon, increasing the

chance of an AdvFS buffer cache hit, and improving the consolidation of I/O requests. By default, smooth sync is enabled on your system.

AdvFS uses I/O request queues to cache asynchronous I/O requests before they are handed to the device driver. Without smooth sync enabled, every 30 seconds, the `update` daemon flushes data from memory to disk, regardless of how long a buffer has been cached. However, with smooth sync enabled (the default), the `update` daemon will not automatically flush the AdvFS ready queue buffers. Instead, asynchronous I/O requests remain in the smooth sync queue for the amount of time specified by the value of the `vfs` attribute `smoothsync_age` (the default is 30 seconds). After this time, the buffer moves to the ready queue.

You enable smooth sync functionality (the default) by using the `smoothsync_age` attribute. However, you do not specify a value for `smoothsync_age` in the `/etc/sysconfigtab` file. Instead, the `/etc/inittab` file is used to enable smooth sync when the system boots to multiuser mode, and to disable smooth sync when the system goes from multiuser mode to single-user mode. This procedure is necessary to reflect the behavior of the `update` daemon, which operates only in multiuser mode.

To enable smooth sync, the following lines must be included in the `/etc/inittab` file and the time limit for caching buffers in the smooth sync queue must be specified (the default is 30 seconds):

```
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
```

Performance Benefit and Tradeoff

Increasing the amount of time an asynchronous I/O request remains in the smooth sync queue increases the chance that a buffer cache hit will occur, which improves AdvFS performance if data is reused. However, this also increases the chance that data may be lost if a system crash occurs.

Decreasing the value of the `smoothsync_age` attribute will speed the flushing of buffers.

When to Tune

You may want to increase the amount of time an asynchronous I/O request remains in the smooth sync queue if you reuse AdvFS data.

Recommended Values

Thirty seconds is the default smooth sync queue timeout limit. If you increase the value of the `smoothsync_age` attribute in the `/etc/inittab` file, you may improve the chance of a buffer cache hit by retaining buffers on the smooth sync queue for a longer period of time. Use the `advfsstat -S` command to show the AdvFS smooth sync queue statistics.

To disable smooth sync, specify a value of 0 (zero) for the `smoothsync_age` attribute.

9.3.6.6 Specifying the Maximum Number of I/O Requests on the Device Queue

Small, logically contiguous AdvFS I/O requests are consolidated into larger I/O requests and put on the device queue, before they are sent to the device driver. See Section 9.3.2 for more information about AdvFS queues.

The `AdvfsMaxDevQLen` attribute controls the maximum number of I/O requests on the device queue. When the number of requests on the queue exceeds this value, only synchronous requests are accepted onto the device queue.

Performance Benefit and Tradeoff

Increasing the size of the device queue can keep devices busy, but may degrade response time.

Decreasing the size of the device queue decreases the amount of time it takes to complete a synchronous (blocking) I/O operation and can improve response time.

You can modify the `AdvfsMaxDevQLen` attribute without rebooting the system.

When to Tune

Although the default value of the `AdvfsMaxDevQLen` attribute is appropriate for many configurations, you may need to modify this value. Increase the default value of the `AdvfsMaxDevQLen` attribute only if devices are not being kept busy.

Recommended Values

The default value of the `AdvfsMaxDevQLen` attribute is 24 requests. The minimum value is 0; the maximum value is 65536. A guideline is to specify a value for the `AdvfsMaxDevQLen` attribute that is less than or equal to the average number of I/O operations that can be performed in 0.5 seconds.

Make sure that increasing the size of the device queue does not cause a decrease in response time. To calculate response time, multiply the value of the `AdvfsMaxDevQLen` attribute by the average I/O latency time for your disks.

If you do not want to limit the number of requests on the device queue, set the value of the `AdvfsMaxDevQLen` attribute to 0 (zero), although this is not recommended.

See Section 3.6 for information about modifying kernel subsystem attributes.

9.3.6.7 Disabling the Flushing of Modified mmapped Pages

The AdvFS buffer cache can contain modified data due to a `write` system call or a memory write reference after an `mmap` system call. The `update` daemon runs every 30 seconds and issues a `sync` call for every fileset mounted with read and write access. However, if smooth sync is enabled (the default), the `update` daemon will not flush the ready queue. Instead, asynchronous I/O requests remain in the smooth sync queue for the amount of time specified by the value of the `vfs` attribute `smoothsync_age` (the default is 30 seconds). See Section 9.3.6.5 for information about the smooth sync queue.

The `AdvfsSyncMmapPages` attribute controls whether modified (dirty) mmapped pages are flushed to disk during a `sync` system call. If the `AdvfsSyncMmapPages` attribute is set to 1 (the default), the modified mmapped pages are asynchronously written to disk. If the `AdvfsSyncMmapPages` attribute is set to 0, modified mmapped pages are not written to disk during a `sync` system call.

Performance Benefit

Disabling the flushing of modified mmapped pages may improve performance of applications that manage their own `mmap` page flushing.

You can modify the `AdvfsSyncMmapPages` attribute without rebooting the system.

When to Tune

Disable flushing mmapped pages only if your applications manage their own `mmap` page flushing.

Recommended Values

If your applications manage their own `mmap` page flushing, set the value of the `AdvfsSyncMmapPages` attribute to zero.

See `mmap`(2) and `msync`(2) for more information. See Section 3.6 for information about modifying kernel subsystem attributes.

9.3.7 Improving AdvFS Performance

After you configure AdvFS, as described in Section 9.3.4, you may be able to improve performance by performing some administrative tasks.

Table 9–4 lists AdvFS performance improvement guidelines and performance benefits as well as tradeoffs.

Table 9–5: AdvFS Performance Improvement Guidelines

Guideline	Performance Benefit	Tradeoff
Defragment file domains (Section 9.3.7.1)	Improves read and write performance	Procedure is time-consuming
Decrease the I/O transfer read-ahead size (Section 9.3.7.2)	Improves performance for `mmap` page faulting	None
Move the transaction log to a fast or uncongested volume (Section 9.3.7.3)	Prevents log from becoming a bottleneck	None
Balance files across volumes in a file domain (Section 9.3.7.4)	Improves performance and evens the future distribution of files	None
Migrate frequently used or large files to different file domains (Section 9.3.7.5)	Improves I/O performance	None

The following sections describe the AdvFS performance improvement guidelines in detail.

9.3.7.1 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. In addition, if a domain has a large number of small files, you may prematurely run out of disk space due to fragmentation.

Use the `defragment` utility to reduce the amount of file fragmentation in a file domain by attempting to make the files more contiguous, which reduces the number of file extents. The utility does not affect data availability and is transparent to users and applications. Striped files are not defragmented.

Performance Benefit and Tradeoff

Defragmenting improves AdvFS performance by making AdvFS disk I/O more efficient. However, the defragment process can be time-consuming and requires disk space in order to run.

When to Perform this Task

Compaq recommends that you run defragment only if you experience problems because of excessive fragmentation and only when there is low file system activity. In addition, there is little performance benefit from defragmenting in the following circumstances:

- A file domain contains primarily files that are smaller than 8 KB.
- A file domain is used in a mail server.
- A file domain is read-only.

To determine if a file domain is fragmented, use the defragment utility with the –v and –n options to show the amount of file fragmentation. Ideally, you want few extents for each file. For example:

```
# defragment -vn staff_dmn

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
```

You can also use the showfile command to check a file's fragmentation. See Section 9.3.5.4 for information.

Recommended Procedure

You can improve the efficiency of the defragmenting process by deleting any unneeded files in the file domain before running the defragment utility. See defragment(8) for more information.

9.3.7.2 Decreasing the I/O Transfer Size

AdvFS reads and writes data by a fixed number of 512-byte blocks. The default value depends on the disk driver's reported preferred transfer size. For example, a common default value is either 128 blocks or 256 blocks.

If you use the addvol or mkfdmn command on a Logical Storage Manager (LSM) volume, the preferred transfer size may be larger than if LSM was not used. The value depends on how you configured the LSM volume.

Performance Benefit

You may be able to improve performance for mmap page faulting and reduce read-ahead paging and cache dilution by decreasing the read-ahead size.

When to Perform this Task

You may want to decrease the I/O transfer size if you experience performance problems with AdvFS I/O throughput.

Recommended Procedure

To display the range of I/O transfer sizes, use the `chvol -l` command. Use the `chvol -r` command to modify the read I/O transfer size (the amount of data read for each I/O request). Use the `chvol -w` command to modify the write I/O transfer size (the amount of data written for each I/O request).

You can decrease the read-ahead size by using the `chvol -r` command.

You can decrease the amount of data written for each I/O request by using the `chvol -w` command. In general, you want to maximize the amount of data written for each I/O by using the default write I/O transfer size or a larger value.

However, in some cases (for example, if you are using LSM volumes), you may need to reduce the AdvFS write-consolidation size. If your AdvFS domains are using LSM, the default preferred transfer size is high, and I/O throughput is not optimal, reduce the write I/O transfer size.

See `chvol`(8) for more information.

9.3.7.3 Moving the Transaction Log

The AdvFS transaction log should be located on a fast or uncongested disk and bus; otherwise, performance may be degraded.

Performance Benefit

Locating the transaction log on a fast or uncongested bus improves performance.

When to Tune

Use the `showfdmn` command to determine the current location of the transaction log. In the `showfdmn` command display, the letter `L` displays next to the volume that contains the log. Move the transaction log if the volume on which it resides is busy and the transaction log is a bottleneck. See `showfdmn`(8) for more information.

Recommended Procedure

Use the `switchlog` command to relocate the transaction log of the specified file domain to a faster or less congested volume in the same domain. See `switchlog`(8) for more information.

In addition, you can divide a large multi-volume file domain into several smaller file domains. This will distribute the transaction log I/O across multiple logs.

9.3.7.4 Balancing a Multivolume File Domain

If the files in a multivolume domain are not evenly distributed, performance may be degraded. Use the `balance` utility to distribute the percentage of used space evenly across volumes in a multivolume file domain. This improves performance and the distribution of future file allocations. 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.

The `balance` utility does not affect data availability and is transparent to users and applications. If possible, use the `defragment` utility before you balance files.

The `balance` utility does not generally split files. Therefore, file domains with very large files may not balance as evenly as file domains with smaller files.

Performance Benefit

Balancing files across the volumes in a file domain improves the distribution of disk I/O.

When to Perform this Task

You may want to balance a file domain if the files are not evenly distributed across the domain.

To determine if you need to balance your files across volumes, use the `showfdmn` command to display information about the volumes in a domain. The `% Used` field shows the percentage of volume space that is currently allocated to files or metadata (fileset data structure). In the following example, the `usr_domain` file domain is not balanced. Volume 1 has 63% used space while volume 2 has 0% used space (it has just been added).

```
# showfdmn usr_domain

               Id     Date Created       LogPgs Version   Domain Name
3437d34d.000ca710  Sun Oct 5 10:50:05 1997  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%
```

See `showfdmn`(8) for more information.

Recommended Procedure

Use the `balance` utility to distribute the percentage of used space evenly across volumes in a multivolume file domain. See `balance`(8) for more information.

9.3.7.5 Migrating Files Within a File Domain

Performance may degrade if too many frequently accessed or large files reside on the same volume in a multivolume file domain. You can improve I/O performance by altering the way files are mapped on the disk.

Use the `migrate` utility to move frequently accessed or large files to different volumes in the file domain. You can specify the volume where a file is to be moved, or allow the system to pick the best space in the file domain. You can migrate either an entire file or specific pages to a different volume.

In addition, the `migrate` command enables you to defragment a specific file and make the file more contiguous, which improves performance.

Performance Benefit

Distributing the I/O load across the volumes in a file domain improves AdvFS performance.

When to Perform this Task

To determine which files to move, use the `showfile -x` command to look at the extent map and the performance percentage of a file. A low performance percentage (less than 80%) indicates that the file is fragmented on the disk. The extent map shows whether the entire file or a portion of the file is fragmented.

The following example displays the extent map of a file called `src`. The file, which resides in a two-volume file domain, shows an 18% performance efficiency in the `Perf` field.

```
# showfile -x src

    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 `src` consists of 11 file extents. This file would be a good candidate to move to another volume to reduce the number of file extents.

See Section 8.2 for information about using commands to determine if file system I/O is evenly distributed.

Recommended Procedure

Use the `migrate` utility to move frequently accessed or large files to different volumes in the file domain. Note that using the `balance` utility after migrating files may cause the files to move to a different volume.

See `migrate`(8) and `balance`(8) for more information.

9.4 Managing UFS Performance

The UNIX File System (UFS) can provide you with high-performance file system operations, especially for critical applications. For example, UFS file reads from striped disks can be 50 percent faster than if you are using AdvFS, and will consume only 20 percent of the CPU power that AdvFS requires.

However, unlike AdvFS, the UFS file system directory hierarchy is bound tightly to a single disk partition.

The following sections describe:

- Using the UFS guidelines to set up a high-performance configuration (Section 9.4.1)
- Obtaining information about UFS performance (Section 9.4.2)
- Tuning UFS in order to improve performance (Section 9.4.3)

9.4.1 UFS Configuration Guidelines

There are a number of parameters that can improve the UFS performance. You can set all of the parameters when you use the `newfs` command to create a file system. For existing file systems, you can modify some parameters by using the `tunefs` command. See `newfs`(8) and `tunefs`(8) for more information.

Table 9–6 describes UFS configuration guidelines and performance benefits as well as tradeoffs.

Table 9–6: UFS Configuration Guidelines

Guideline	Performance Benefit	Tradeoff
Make the file system fragment size equal to the block size (Section 9.4.1.1)	Improves performance for large files	Wastes disk space for small files

Table 9–6: UFS Configuration Guidelines (cont.)

Guideline	Performance Benefit	Tradeoff
Use the default file system fragment size of 1 KB (Section 9.4.1.1)	Uses disk space efficiently	Increases the overhead for large files
Reduce the density of inodes on a file system (Section 9.4.1.2)	Frees disk space for file data and improves large file performance	Reduces the number of files that can be created on the file system
Allocate blocks sequentially (Section 9.4.1.3)	Improves performance for disks that do not have a read-ahead cache	Reduces the total available disk space
Increase the number of blocks combined for a cluster (Section 9.4.1.4)	May decrease number of disk I/O operations	May require more memory to buffer data
Use a Memory File System (MFS) (Section 9.4.1.5)	Improves I/O performance	Does not ensure data integrity because of cache volatility
Use disk quotas (Section 9.4.1.6)	Controls disk space utilization	UFS quotas may result in a slight increase in reboot time
Increase the maximum number of UFS and MFS mounts (Section 9.4.1.7)	Allows more mounted file systems	Requires additional memory resources

The following sections describe the UFS configuration guidelines in detail.

9.4.1.1 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 so that it is 25, 50, 75, or 100 percent of the block size.

The UFS file system block size can be 8 KB (the default), 16 KB, 32 KB, or 64 KB. The default fragment size is 1 KB. You can modify the fragment size so that it is 25, 50, 75, or 100 percent of the block size. Use the `newfs` command to modify block and fragment sizes.

Although the default fragment size uses disk space efficiently, it increases the overhead for large files. If the average file in a file system is larger than 16 KB but less than 96 KB, you may be able to 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.

9.4.1.2 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 will reduce 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. See `newfs`(8) for more information.

9.4.1.3 Allocating Blocks Sequentially

The UFS `rotdelay` parameter specifies the time, in milliseconds, to service a transfer completion interrupt and initiate a new transfer on the same disk. You can set the `rotdelay` parameter to 0 (the default) to allocate blocks sequentially. This is useful for disks that do not have a read-ahead cache. However, it will reduce the total amount of available disk space.

Use either the `tunefs` command or the `newfs` command to modify the `rotdelay` value. See `newfs`(8) and `tunefs`(8) for more information.

9.4.1.4 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. However, more memory will be needed to cache data.

Use the `tunefs` command or the `newfs` command to change the value of `maxcontig`. See `newfs`(8) and `tunefs`(8) for more information.

9.4.1.5 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 information.

9.4.1.6 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. For information about AdvFS quotas, see Section 9.3.4.10. For information about UFS quotas, see the *System Administration* manual.

9.4.1.7 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. The `vfs` subsystem attribute `max_ufs_mounts` specifies the maximum number of UFS and MFS mounts on the system.

Performance Benefit and Tradeoff

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.

You can modify the `max_ufs_mounts` attribute without rebooting the system.

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.

Recommended Values

The default value of the max_ufs_mounts attribute is 1000.

See Section 3.6 for information about modifying kernel subsystem attributes.

9.4.2 Gathering UFS Information

Table 9–7 describes the tools you can use to obtain information about UFS.

Table 9–7: UFS Monitoring Tools

Name	Use	Description
dumpfs	Displays UFS information (Section 9.4.2.1)	Displays detailed information about a UFS file system or a special device, including information about the file system fragment size, the percentage of free space, super blocks, and the cylinder groups.
(dbx) print ufs_clusterstats	Reports UFS clustering statistics (Section 9.4.2.2)	Reports statistics on how the system is performing cluster read and write transfers.
(dbx) print bio_stats	Reports UFS metadata buffer cache statistics (Section 9.4.2.3)	Reports statistics on the metadata buffer cache, including superblocks, inodes, indirect blocks, directory blocks, and cylinder group summaries.

The following sections describe these commands in detail.

9.4.2.1 Displaying UFS Information by Using the dumpfs Command

The dumpfs command displays UFS information, including super block and cylinder group information, for a specified file system. Use this command to obtain information about the file system fragment size and the minimum free space percentage. The following example shows part of the output of the dumpfs command:

```
# /usr/sbin/dumpfs /devices/disk/dsk0g | more
magic   11954   format  dynamic time   Tue Sep 14 15:46:52 1998
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 zero, then blocks are allocated contiguously.

9.4.2.2 Monitoring UFS Clustering by Using the dbx Debugger

To determine how efficiently the system is performing cluster read and write transfers, use the `dbx print` command to examine the `ufs_clusterstats` data structure.

The following example shows a system that is not clustering efficiently:

```
# /usr/ucb/dbx -k /vmunix /dev/mem
(dbx) print ufs_clusterstats
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
    }
}
(dbx)
```

The preceding example shows 24644 single-block transfers and no 9-block transfers. A single block is 8 KB. The trend of the data shown in the example is the reverse of what you want to see. It shows a large number of single-block transfers and a declining number of multiblock (1–9) transfers. However, if the files are all small, this may be the best blocking that you can achieve.

You can examine the cluster reads and writes separately with the `ufs_clusterstats_read` and `ufs_clusterstats_write` data structures.

See Section 9.4.3 for information on tuning UFS.

9.4.2.3 Checking the Metadata Buffer Cache by Using the dbx Debugger

The metadata buffer cache contains UFS file metadata—superblocks, inodes, indirect blocks, directory blocks, and cylinder group summaries. To check the metadata buffer cache, use the `dbx print` command to examine the `bio_stats` data structure.

Consider the following example:

```
# /usr/ucb/dbx -k /vmunix /dev/mem
(dbx) print bio_stats
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
}
(dbx)
```

If the miss rate is high, you may want to raise the value of the `bufcache` attribute. 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.

See Section 9.4.3.1 for information on how to tune the metadata buffer cache.

9.4.3 Tuning UFS

After you configure your UFS file systems, you may be able to improve UFS performance. To successfully improve performance, you must understand how your applications and users perform file system I/O, as described in Section 2.1.

Table 9–8 describes UFS tuning guidelines and performance benefits as well as tradeoffs. The guidelines described in Table 9–1 also apply to UFS configurations.

Table 9–8: UFS Tuning Guidelines

Guideline	Performance Benefit	Tradeoff
Increase the size of metadata buffer cache to more than 3 percent of main memory (Section 9.4.3.1)	Increases cache hit rate and improves UFS performance	Requires additional memory resources
Increase the size of the metadata hash chain table (Section 9.4.3.2)	Improves UFS lookup speed	Increases wired memory
Increase the smooth sync caching threshold for asynchronous UFS I/O requests (Section 9.4.3.3)	Improves performance of AdvFS asynchronous I/O	Increases the chance that data may be lost if a system crash occurs
Delay flushing UFS clusters to disk (Section 9.4.3.4)	Frees CPU cycles and reduces number of I/O operations	May degrade real-time workload performance when buffers are flushed
Increase number of blocks combined for read ahead (Section 9.4.3.5)	May reduce disk I/O operations	May require more memory to buffer data
Increase number of blocks combined for a cluster (Section 9.4.3.6)	May decrease disk I/O operations	Reduces available disk space
Defragment the file system (Section 9.4.3.7)	Improves read and write performance	Requires down time

The following sections describe how to tune UFS in detail.

9.4.3.1 Increasing the Size of the Metadata Buffer Cache

At boot time, the kernel wires a percentage of physical memory for the metadata buffer cache, which temporarily holds recently accessed UFS and CD-ROM File System (CDFS) metadata. The `vfs` subsystem attribute `bufcache` specifies the size of the metadata buffer cache as a percentage of physical memory. See Section 6.1.2.1 for information about how memory is allocated to the metadata buffer cache.

Performance Benefit and Tradeoff

Allocating additional memory to the metadata buffer cache may improve UFS performance if you reuse files, but it will reduce the amount of memory available to processes and the UBC.

You cannot modify the `bufcache` attribute without rebooting the system.

When to Tune

Usually, you do not have to increase the size of the metadata buffer cache.

However, you may want to increase the size of the cache if you reuse data and have a high cache miss rate (low hit rate). To determine whether to increase the size of the metadata buffer cache, use the `dbx print` command to examine 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, you may want to increase the cache size. See Section 9.4.2.3 for more information.

Recommended Values

The default value of the `bufcache` attribute is 3 percent.

If you have a general-purpose timesharing system, do not increase the value of the `bufcache` attribute to more than 10 percent. If you have an NFS server that does not perform timesharing, do not increase the value of the `bufcache` attribute to more than 35 percent.

See Section 3.6 for information about modifying kernel subsystem attributes.

9.4.3.2 Increasing the Size of the Metadata Hash Chain Table

The hash chain table for the metadata buffer cache stores the heads of the hashed buffer queues. The `vfs` subsystem attribute `buffer_hash_size` specifies the size of the hash chain table, in table entries, for the metadata buffer cache.

Performance Benefit and Tradeoff

Increasing the size of the hash chain table distributes the buffers, which makes the average chain lengths short. This can improve lookup speeds. However, increasing the size of the hash chain table increases wired memory.

You cannot modify the `buffer_hash_size` attribute without rebooting the system.

When to Tune

Usually, you do not have to modify the size of the hash chain table.

Recommended Values

The minimum size of the buffer_hash_size attribute is 16; the maximum size is 524287. The default value is 512.

You can modify the value of the buffer_hash_size attribute so that each hash chain has 3 or 4 buffers. To determine a value for the buffer_hash_size attribute, use the dbx print command to examine the value of the nbuf kernel variable, then divide the value by 3 or 4, and finally round the result to a power of 2. For example, if nbuf has a value of 360, dividing 360 by 3 gives you a value of 120. Based on this calculation, specify 128 (2 to the power of 7) as the value of the buffer_hash_size attribute.

See Section 3.6 for information about modifying kernel attributes.

9.4.3.3 Increasing the UFS Smooth Sync Cache Timeout Value

Smooth sync functionality improves UFS I/O performance by preventing I/O spikes caused by the update daemon, and by increasing the UBC hit rate, which decreases the total number of disk operations. Smooth sync also helps to efficiently distribute I/O requests over the sync interval, which decreases the length of the disk queue and reduces the latency that results from waiting for a busy page to be freed. By default, smooth sync is enabled on your system.

UFS caches asynchronous I/O requests in the dirty-block queue and in the UBC object dirty-page list queue before they are handed to the device driver. With smooth sync enabled (the default), the update daemon will not flush buffers from the dirty page lists and dirty wired page lists. Instead, the buffers get moved to the device queue only after the amount of time specified by the value of the vfs attribute smoothsync_age (the default is 30 seconds). After this time, the buffer moves to the device queue.

If smooth sync is disabled, every 30 seconds the update daemon flushes data from memory to disk, regardless of how long a buffer has been cached.

Smooth sync functionality is controlled by the smoothsync_age attribute. However, you do not specify a value for smoothsync_age in the /etc/sysconfigtab file. Instead, the /etc/inittab file is used to enable smooth sync when the system boots to multiuser mode and to disable smooth sync when the system goes from multiuser mode to single-user mode. This procedure is necessary to reflect the behavior of the update daemon, which operates only in multiuser mode.

To enable smooth sync, the following lines must be included in the /etc/inittab file and the time limit for caching buffers in the smooth sync queue must be specified (default is 30 seconds):

```
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
```

Performance Benefit and Tradeoff

Increasing the amount of time that an asynchronous I/O request ages before being placed on the device queue (increasing the value of the smoothsync_age attribute) will increase the chance that a buffer cache hit will occur, which improves UFS performance if the data is reused. However, this increases the chance that data may be lost if a system crash occurs.

Decreasing the value of the smoothsync_age attribute will speed the flushing of buffers.

When to Tune

Usually, you do not have to modify the smooth sync queue timeout limit.

Recommended Values

Thirty seconds is the default smooth sync queue timeout limit. If you increase the value of the smoothsync_age attribute in the /etc/inittab file, you will increase the chance that a buffer cache hit will occur.

To disable smooth sync, specify a value of 0 (zero) for the smoothsync_age attribute.

See Section 3.6 for information about modifying kernel subsystem attributes.

9.4.3.4 Delaying UFS Cluster Flushing

By default, clusters of UFS pages are written asynchronously (the write must be completed). Enabling the delay_wbuffers kernel variable causes these clusters to be written synchronously (delayed), as other dirty data and metadata pages are written. However, if the percentage of UBC dirty pages reaches the value of the delay_wbuffers_percent kernel variable, the clusters will be written asynchronously, regardless of the setting of the delay_wbuffers kernel variable.

Performance Benefit and Tradeoff

Delaying full write buffer flushing can free CPU cycles. However, it may adversely affect real-time workload performance, because the system will experience a heavy I/O load at sync time.

You can modify the delay_wbuffers kernel variable without rebooting the system.

When to Tune

Delay cluster flushing if your applications frequently write to previously written pages. This can result in a net decrease in the total number of I/O requests.

Recommended Values

To delay cluster flushing, use the `dbx patch` command to set the value of the `delay_wbuffers` kernel variable to 1 (enabled). The default value of `delay_wbuffers` is 0 (disabled).

See Section 3.6.7 for information on using `dbx`.

9.4.3.5 Increasing the Number of Blocks Combined for Read-Ahead

You can increase the number of blocks that are combined for a read-ahead operation.

Performance Benefit and Tradeoff

Increase the number of blocks combined for read-ahead if your applications can use a large read-ahead size.

When to Tune

Usually, you do not have to increase the number of blocks combined for read-ahead.

Recommended Values

To increase the number of blocks combined for read-ahead, use the `dbx patch` command to set the value of the `cluster_consec_init` kernel variable equal to the value of the `cluster_max_read_ahead` kernel variable (the default is 8), which specifies the maximum number of read-ahead clusters that the kernel can schedule.

In addition, you must make sure that cluster read operations are enabled on nonread-ahead and read-ahead blocks. To do this, use `dbx` to set the value of the `cluster_read_all` kernel variable to 1, which is the default value.

See Section 3.6.7 for information on using `dbx`.

9.4.3.6 Increasing the Number of Blocks Combined for a Cluster

You can increase the number of blocks combined for a cluster. The `cluster_maxcontig` kernel variable specifies the number of blocks that are combined into a single I/O operation. Contiguous writes are done in a unit size that is determined by the file system block size (8 KB) multiplied by the value of the `cluster_maxcontig` parameter.

Performance Benefit and Tradeoff

Increase the number of blocks combined for a cluster if your applications can use a large cluster size.

When to Tune

Usually, you do not have to increase the number of blocks combined for a cluster.

Recommended Values

The default value of `cluster_maxcontig` kernel variable is 8.

See Section 3.6.7 for information about using `dbx`.

9.4.3.7 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.

Performance Benefit and Tradeoff

Defragmenting a UFS file system improves file system performance. However, it is a time-consuming process.

When to Perform This Task

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 9.4.2.2 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.

9.5 Managing NFS Performance

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.

You can monitor NFS by using the `nfsstat` and other commands. 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. See `nfsstat`(8) for more information.

The following sections describe how to perform the following tasks:

- Gather NFS performance information (Section 9.5.1)
- Improving NFS performance (Section 9.5.2)

9.5.1 Gathering NFS Information

Table 9–9 describes the commands you can use to obtain information about NFS operations.

Table 9–9: NFS Monitoring Tools

Name	Use	Description
`nfsstat`	Displays network and NFS statistics (Section 9.5.1.1)	Displays NFS and 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`).

Table 9–9: NFS Monitoring Tools (cont.)

Name	Use	Description
nfswatch	Monitors an NFS server	Monitors all incoming network traffic to an NFS server and divides it into several categories, including NFS reads and writes, NIS requests, and RPC authorizations. Your kernel must be configured with the packetfilter option to use the command. See nfswatch(8) and packetfilter(7) for more information.
ps axlmp	Displays information about idle threads (Section 9.5.1.2)	Displays information about idle threads on a client system.
(dbx) print nfs_sv_active_hist	Displays active NFS server threads (Section 3.6.7)	Displays a histogram of the number of active NFS server threads.
(dbx) print nchstats	Displays the hit rate (Section 9.1.2)	Displays the namei cache hit rate.
(dbx) print bio_stats	Displays metadata buffer cache information (Section 9.4.2.3)	Reports statistics on the metadata buffer cache hit rate.
(dbx) print vm_tune	Reports UBC statistics (Section 6.3.4)	Reports the UBC hit rate.

The following sections describe how to use some of these tools.

9.5.1.1 Displaying NFS Information by Using the nfsstat Command

The nfsstat command displays statistical information about NFS and Remote Procedure Call (RPC) interfaces in the kernel. You can also use this command to reinitialize the statistics.

An example of the nfsstat command is as follows:

```
# /usr/ucb/nfsstat

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.

Use the `nfsstat -s -i 10` command to display NFS and RPC information at ten-second intervals.

If you are attempting to monitor an experimental situation with `nfsstat`, reset the NFS counters to 0 before you begin the experiment. Use the `nfsstat -z` command to clear the counters.

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

9.5.1.2 Displaying Idle Thread Information by Using the ps Command

On a 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 reads 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 most clients, seven threads are sufficient.

The following example uses the ps axlmp command to display idle I/O threads on a client system:

```
#
/usr/ucb/ps axlmp 0 | grep nfs

 0  42   0            nfsiod_  S                0:00.52
 0  42   0            nfsiod_  S                0:01.18
 0  42   0            nfsiod_  S                0:00.36
 0  44   0            nfsiod_  S                0:00.87
 0  42   0            nfsiod_  S                0:00.52
 0  42   0            nfsiod_  S                0:00.45
 0  42   0            nfsiod_  S                0:00.74

#
```

The previous output shows a sufficient number of sleeping threads and 42 server threads that were started by nfsd, where nfsiod_ has been replaced by nfs_tcp or nfs_udp.

If your output shows that few threads are sleeping, you may be able to improve NFS performance by increasing the number of threads. See Section 9.5.2.2, Section 9.5.2.3, nfsiod(8), and nfsd(8) for more information.

9.5.2 Improving NFS Performance

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. 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 in the UBC. If you are using AdvFS, an access structure is also required to keep file data in the UBC.

If you are running NFS over TCP, tuning TCP may improve performance if there are many active clients. See Section 10.2 for more information. However, if you are running NFS over UDP, no network tuning is needed.

Table 9–10 lists NFS tuning and performance-improvement guidelines and the benefits as well as tradeoffs.

Table 9–10: NFS Performance Guidelines

Guideline	Performance Benefit	Tradeoff
Set the value of the `maxusers` attribute to the number of server NFS operations that are expected to occur each second (Section 5.1)	Provides the appropriate level of system resources	Consumes memory
Increase the size of the namei cache (Section 9.2.1)	Improves file system performance	Consumes memory
Increase the number of AdvFS access structures, if you are using AdvFS (Section 9.3.6.3)	Improves AdvFS performance	Consumes memory
Increase the size of the metadata buffer cache, if you are using UFS (Section 9.4.3.1)	Improves UFS performance	Consumes wired memory
Use Prestoserve (Section 9.5.2.1)	Improves synchronous write performance for NFS servers	Cost
Configure the appropriate number of threads on an NFS server (Section 9.5.2.2)	Enables efficient I/O blocking operations	None
Configure the appropriate number of threads on the client system (Section 9.5.2.3)	Enables efficient I/O blocking operations	None
Modify cache timeout limits on the client system (Section 9.5.2.4)	May improve network performance for read-only file systems and enable clients to quickly detect changes	Increases network traffic to server
Decrease network timeouts on the client system (Section 9.5.2.5)	May improve performance for slow or congested networks	Reduces the theoretical performance
Use NFS Protocol Version 3 on the client system (Section 9.5.2.6)	Improves network performance	Decreases the performance benefit of Prestoserve

The following sections describe some of these guidelines.

9.5.2.1 Using Prestoserve to Improve NFS Server Performance

You can improve NFS performance by installing Prestoserve on the server. Prestoserve greatly improves synchronous write performance for servers that are using NFS Version 2. Prestoserve enables an NFS Version 2 server to write client data to a nonvolatile (battery-backed) cache, instead of writing the data to disk.

Prestoserve may improve write performance for NFS Version 3 servers, but not as much as with NFS Version 2, because NFS Version 3 servers can reliably write data to volatile storage without risking loss of data in the event of failure. NFS Version 3 clients can detect server failures and resend any write data that the server may have lost in volatile storage.

See the *Guide to Prestoserve* for more information.

9.5.2.2 Configuring Server Threads

The `nfsd` daemon runs on NFS servers to service NFS requests from client machines. The daemon spawns a number of server threads that process NFS requests from client machines. 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. See `nfsd`(8) for more information.

9.5.2.3 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 or 15 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 any 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 zero. See Section 3.6.7 for information.

Use the `ps axlmp 0 | grep nfs` command to display idle I/O threads on the client. If few threads are sleeping, you may be able to improve NFS performance by increasing the number of threads. See `nfsiod`(8) for more information.

9.5.2.4 Modifying Cache Timeout Limits

For read-only file systems and slow network links, performance may be improved by changing the cache timeout limits on NFS client systems. These timeouts affect how quickly you see updates to a file or directory that has been 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.

9.5.2.5 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 9.5.1.1 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 10.1.1 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.

9.5.2.6 Using NFS Protocol Version 3

NFS Protocol Version 3 provides NFS client-side asynchronous write support, which improves the cache consistency protocol and requires less network load than Version 2. These performance improvements slightly decrease the performance benefit that Prestoserve provided for NFS Version 2. However, with Protocol Version 3, Prestoserve still speeds file creation and deletion.

10

Managing Network Performance

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

Name	Use	Description
`netstat`	Displays network statistics (Section 10.1.1)	Displays 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.
`traceroute`	Displays the packet route to a network host	Tracks the route network packets follow from gateway to gateway. See `traceroute`(8) for more information.
`ping`	Determines if a system can be reached on the network	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) for more information.
`sobacklog_hiwat` attribute	Reports the maximum number of pending requests to any server socket (Section 10.1.2)	Allows you to display the maximum number of pending requests to any server socket in the system.

Table 10–1: Network Monitoring Tools (cont.)

Name	Use	Description
sobacklog_drops attribute	Reports the number of backlog drops that exceed a socket backlog limit (Section 10.1.2)	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.
somaxconn_drops attribute	Reports the number of drops that exceed the value of the somaxconn attribute (Section 10.1.2)	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).
tcpdump	Monitors network interface packets	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 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 tcpdump(8) and packetfilter(7) for more information.

The following sections describe some of these commands in detail.

10.1.1 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.
- Use the netstat -is command to check for network device driver errors.

- 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.

- 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.

  An example of the `netstat -an` command is as follows:

```
# /usr/sbin/netstat -an | grep tcp | awk '{print $6}' | sort | uniq -c
     1 CLOSE_WAIT
    58 ESTABLISHED
     2 FIN_WAIT_1
     3 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. In the previous example, there are almost 16,000 sockets being used, which requires 16 MB of memory.

- Use the `netstat -p ip` command to check for bad checksums, length problems, excessive redirects, and packets lost because of resource problems.

- Use the `netstat -p tcp` command to check for retransmissions, out of order packets, and bad checksums.

- Use the `netstat -p udp` command to check for bad checksums and full sockets.

- Use the `netstat -rs` command to obtain routing statistics.

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* manual for more information on how to diagnose failures.

The following output produced by the `netstat -i` command shows input and output errors:

```
# /usr/sbin/netstat -i
Name  Mtu   Network     Address         Ipkts Ierrs    Opkts Oerrs  Coll
ln0   1500  DLI         none           133194     2    23632     4  4881
ln0   1500  <Link>                     133194     2    23632     4  4881
ln0   1500  red-net     node1          133194     2    23632     4  4881
sl0*  296   <Link>                          0     0        0     0     0
sl1*  296   <Link>                          0     0        0     0     0
lo0   1536  <Link>                       580     0      580     0     0
lo0   1536  loop        localhost        580     0      580     0     0
```

Use the following netstat command to determine the causes of the input (Ierrs) and output (Oerrs) shown in the preceding example:

```
# /usr/sbin/netstat -is

ln0 Ethernet counters at Fri Jan 14 16:57:36 1998

        4112 seconds since last zeroed
    30307093 bytes received
     3722308 bytes sent
      133245 data blocks received
       23643 data blocks sent
    14956647 multicast bytes received
      102675 multicast blocks received
       18066 multicast bytes sent
         309 multicast blocks sent
        3446 blocks sent, initially deferred
        1130 blocks sent, single collision
        1876 blocks sent, multiple collisions
           4 send failures, reasons include:
                Excessive collisions
           0 collision detect check failure
           2 receive failures, reasons include:
                Block check error
                Framing Error
           0 unrecognized frame destination
           0 data overruns
           0 system buffer unavailable
           0 user buffer unavailable
```

The following netstat -s displays statistics for each protocol:

```
# /usr/sbin/netstat -s
ip:
        67673 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
        8616 fragments received
        0 fragments dropped (dup or out of space)
        5 fragments dropped after timeout
        0 packets forwarded
        8 packets not forwardable
        0 redirects sent
icmp:
        27 calls to icmp_error
        0 errors not generated  old message was icmp
        Output histogram:
                echo reply: 8
                destination unreachable: 27
        0 messages with bad code fields
        0 messages < minimum length
        0 bad checksums
        0 messages with bad length
        Input histogram:
```

```
                echo reply: 1
                destination unreachable: 4
                echo: 8
        8 message responses generated
igmp:
        365 messages received
        0 messages received with too few bytes
        0 messages received with bad checksum
        365 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:
        11219 packets sent
                7265 data packets (139886 bytes)
                4 data packets (15 bytes) retransmitted
                3353 ack-only packets (2842 delayed)
                0 URG only packets
                14 window probe packets
                526 window update packets
                57 control packets
        12158 packets received
                7206 acks (for 139930 bytes)
                32 duplicate acks
                0 acks for unsent data
                8815 packets (1612505 bytes) received in-sequence
                432 completely duplicate packets (435 bytes)
                0 packets with some dup. data (0 bytes duped)
                14 out-of-order packets (0 bytes)
                1 packet (0 bytes) of data after window
                0 window probes
                1 window update packet
                5 packets received after close
                0 discarded for bad checksums
                0 discarded for bad header offset fields
                0 discarded because packet too short
        19 connection requests
        25 connection accepts
        44 connections established (including accepts)
        47 connections closed (including 0 drops)
        3 embryonic connections dropped
        7217 segments updated rtt (of 7222 attempts)
        4 retransmit timeouts
                0 connections dropped by rexmit timeout
        0 persist timeouts
        0 keepalive timeouts
                0 keepalive probes sent
                0 connections dropped by keepalive
udp:
        12003 packets sent
        48193 packets received
        0 incomplete headers
        0 bad data length fields
        0 bad checksums
        0 full sockets
        12943 for no port (12916 broadcasts, 0 multicasts)
```

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

10.1.2 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 zero.

- `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 zero.

- `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 zero.

It is recommended 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* 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

Guideline	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 a PMTU (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

Table 10–2: Network Tuning Guidelines (cont.)

Guideline	Performance Benefit	Tradeoff
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
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 10.2.16)	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 10.2.17)	Helps to prevent dropping UDP packets	May decrease available memory when the buffer space is being used
Allocate sufficient memory to the UBC (Section 9.2.4 and Section 9.2.5 Section 9.2.6)	Improves disk I/O performance	May decrease the physical memory available to processes
Increase the size of the ARP table (Section 10.2.18)	May improve network performance on a system that is simultaneously connected to many nodes on the same LAN	Consumes memory resources
Increase the maximum size of a socket buffer (Section 10.2.19)	Allows large socket buffer sizes	Consumes memory resources
Prevent dropped input packets (Section 10.2.20)	Allows high network loads	None

The following sections describe these tuning guidelines in detail.

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.

See Section 3.6 for information about modifying kernel subsystem attributes.

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.

Compaq recommends 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.

See Section 3.6 for information about modifying kernel attributes.

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 10.1.2 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 zero. 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.

See Section 3.6 for information about modifying kernel subsystem attributes.

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.

It is not recommended 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.

See Section 3.6 for information about modifying kernel subsystem attributes.

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.

See Section 3.6 for information about modifying kernel subsystem attributes.

10.2.6 Disabling Use of a PMTU

Packets transmitted between servers are fragmented into units of a specific size in order 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 a PMTU 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 a PMTU can degrade bandwidth.

You can modify the pmtu_enabled attribute without rebooting the system.

When to Tune

Disable use of a PMTU 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.

Recommended Values

Set the value of the pmtu_enabled attribute to 0 to disable the use of PMTU protocol.

See Section 3.6 for information about modifying kernel subsystem attributes.

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.

It is recommended 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 information.

See Section 3.6 for information about modifying kernel subsystem attributes.

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 (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 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 zero. 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 in order 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 in order to enable keepalive for all sockets.

If you enable keepalive, you can also configure the following TCP options for each socket:

- The `inet` subsystem attribute `tcp_keepidle` specifies the amount of idle time before sending a keepalive probe (specified in 0.5 second units). The default interval is 2 hours.
- The `inet` subsystem attribute `tcp_keepintvl` specifies the amount of time (in 0.5 second units) between the retransmission of keepalive probes. The default interval is 75 seconds.
- The `inet` subsystem attribute `tcp_keepcnt` specifies the maximum number of keepalive probes that are sent before the connection is dropped. The default is 8 probes.
- The `inet` subsystem attribute `tcp_keepinit` specifies the maximum amount of time before an initial connection attempt times out in 0.5 second units. The default is 75 seconds.

See Section 3.6 for information about modifying kernel subsystem attributes.

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.

See Section 3.6 for information about modifying kernel subsystem attributes.

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).

See Section 3.6 for information about modifying kernel subsystem attributes.

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 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. It is strongly recommended that you use the default value; otherwise, there is the potential for data corruption.

See Section 3.6 for information about modifying kernel subsystem attributes.

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`.

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.

See Section 3.6 for information about modifying kernel subsystem attributes.

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 zero. To disable the TCP acknowledgment delay, set the value of the `tcpnodelack` attribute to one.

See Section 3.6 for information about modifying kernel subsystem attributes.

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.

See Section 3.6 for information about modifying kernel subsystem attributes.

10.2.16 Increasing the Transmit and Receive Buffers for a TCP Socket

The inet subsystem attribute tcp_sendspace specifies the default transmit buffer size for a TCP socket. The tcp_recvspace attribute specifies the default receive buffer size for a TCP socket.

Performance Benefit and Tradeoff

Increasing the transmit and receive socket buffers allows you to buffer more TCP packets per socket. However, increasing the values uses more memory when the buffers are being used by an application (sending or receiving data).

You can modify the tcp_sendspace and tcp_recvspace attributes without rebooting the system.

When to Tune

You may want to increase the transmit and receive socket buffers if you have a busy system with sufficient memory (for example, more than 1 GB of physical memory). Before you apply this modification, you may want to increase the maximum size of a socket buffer, as described in Section 10.2.19.

Recommended Values

The default values of the `tcp_sendspace` and `tcp_recvspace` attributes are 32 KB (32768 bytes). You can increase the value of these attributes to 60 KB.

You may want to increase the maximum size of a socket buffer before you increase the transmit and receive buffers. See Section 10.2.19 for information.

See Section 3.6 for information about modifying kernel subsystem attributes.

10.2.17 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.

See Section 3.6 for information about modifying kernel subsystem attributes.

10.2.18 Increasing the Size of the ARP Table

The `net` subsystem attribute `arptab_nb` specifies the number of hash buckets in the address resolution protocol (ARP) table (that is, the table's width). The `net` subsystem attribute `arptab_depth` specifies the number of entries in each hash bucket in the ARP table.

Performance Benefit and Tradeoff

Increasing the size of the ARP table may improve performance. Wide ARP tables can decrease the chance that a search will be needed to match an address to an ARP entry. Deep ARP tables can hold a large number of entries. However, increasing the size of the ARP table will increase the memory used by the table.

Increasing the size of the ARP table will not affect performance unless the system is simultaneously connected to many nodes on the same LAN. See the *Kernel Debugging* manual and `kdbx`(8) for more information.

You can modify the `arptab_nb` and `arptab_depth` attributes without rebooting the system.

When to Tune

Display the ARP table by using the `arp -a` command or the `kdbx arp` debugger extension. Increase the value of the `arptab_nb` and `arptab_depth` attributes if the ARP table contains more than 400 entries.

Recommended Values

You can increase the width of the ARP table by increasing the value of the `inarptab_nb` attribute. The default value is 37. The maximum value is 1024.

You can increase the depth of the ARP table by increasing the value of the `arptab_depth` attributes. The default value is 16. The maximum value is 256.

See Section 3.6 for information about modifying kernel subsystem attributes.

10.2.19 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 (see Section 10.2.16).

See Section 3.6 for information about modifying kernel subsystem attributes.

10.2.20 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.

See Section 3.6 for information about modifying kernel subsystem attributes.

11

Managing Application Performance

You may be able to improve overall Tru64 UNIX performance by improving application performance. This chapter describes how to:

- Profile and debug applications (Section 11.1)
- Improve application performance (Section 11.2)

11.1 Gathering Profiling and Debugging Information

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

Table 11–1 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 11–1: 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.
`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.

Table 11–1: Application Profiling and Debugging Tools (cont.)

Name	Use	Description
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 11–1: Application Profiling and Debugging Tools (cont.)

Name	Use	Description
prof	Analyzes profiling data and displays a profile of statistics for each procedure in an application	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 11–1: Application Profiling and Debugging Tools (cont.)

Name	Use	Description
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.
`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 is available on the Tru64 UNIX Freeware CD-ROM.

11.2 Improving Application Performance

Well-written applications use CPU, memory, and I/O resources efficiently. Table 11–2 describes some guidelines to improve application performance.

Table 11–2: Application Performance Improvement Guidelines

Guideline	Performance Benefit	Tradeoff
Install the latest operating system patches (Section 11.2.1)	Provides the latest optimizations	None
Use the latest version of the compiler (Section 11.2.2)	Provides the latest optimizations	None
Use parallelism (Section 11.2.3)	Improves SMP performance	None
Optimize applications (Section 11.2.4)	Generates more efficient code	None
Use shared libraries (Section 11.2.5)	Frees memory	May increase execution time
Reduce application memory requirements (Section 11.2.6)	Frees memory	Program may not run optimally
Use memory locking as part of real-time program initialization (Section 11.2.7)	Allows you to lock and unlock memory as needed	Reduces the memory available to processes and the UBC

The following sections describe how to improve application performance.

11.2.1 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. See your customer service representative or for information about installing patches.

11.2.2 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.

11.2.3 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.

11.2.4 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.

11.2.5 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.

11.2.6 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, allocate the data dynamically 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.

11.2.7 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` to determine if a process is locked into memory and cannot be swapped out. See Section 6.3.2.

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* manual and `mlockall`(3) for more information.

Glossary

This glossary lists the terms that are used to describe Tru64 UNIX performance and availability.

active list
Pages that are being used by the virtual memory subsystem or the UBC.

adaptive RAID 3/5
See **dynamic parity RAID**.

anonymous memory
Modifiable memory that is used for stack, heap, or `malloc`.

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. Bandwidth is also called the transfer rate.

bottleneck
A system resource that is being pushed near to its capacity and is causing a performance degradation.

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.

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.

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.

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.

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.

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.

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 RAID 3/5, dynamic parity RAID combines the features of RAID 3 and RAID 5 to improve disk I/O performance and availability for a wide variety of applications. Adaptive RAID 3/5 dynamically adjusts, according to workload needs, between data transfer-intensive algorithms and I/O operation-intensive algorithms.

eager mode

See **immediate mode**.

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.

file-backed memory

Memory that is used for program text or shared libraries.

free list
Pages that are clean and are not being used (the size of this list controls when page reclamation occurs).

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.

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**.

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.

page
The smallest portion of physical memory that the system can allocate (8 KB of memory).

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 that contains 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.

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.

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 software (LSM). Hardware-based RAID functionality is provided by intelligent controllers, caches, disks, and software.

RAID 0
Also known as disk striping, RAID 0 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.

RAID 1

Also known as data mirroring, RAID 1 functionality maintains identical copies of data on different disks in an array. Duplicating data provides high data availability. In addition, RAID 1 improves the disk read performance, because data can be read from two locations. However, RAID 1 decreases disk write performance, because data must be written twice. Mirroring n disks requires $2n$ disks.

RAID 3

RAID 3 functionality divides data blocks and distributes (stripes) the data across a disk array, providing parallel access to data. RAID 3 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. RAID 3 increases bandwidth, but it provides no improvement in the throughput. RAID 3 can improve the I/O performance for applications that transfer large amounts of sequential data.

RAID 5

RAID 5 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. RAID 5 allows independent access to data and can handle simultaneous I/O operations. RAID 5 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 device 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) or the mean time to first failure (MTTF).

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.

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.

rotational latency

The amount of time, in milliseconds, for a disk to rotate to a specific disk sector.

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.

seek time

The amount of time, in milliseconds, for a disk head to move to a specific disk track.

sequential access pattern

Refers to an access pattern in which data is read from or written to contiguous blocks on a disk.

short page fault

A page fault that occurs when a requested address is found in the virtual memory subsystem's internal data structures.

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.

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 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.

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.

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.

tune

To modify the kernel by changing the values of kernel variables, thus improving system performance.

UBC

See **Unified Buffer Cache**.

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 in order 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.

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.

zero-filled-on-demand page fault
A page fault that occurs when a requested address is accessed for the first time.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

How to Order Tru64 UNIX Documentation

You can order documentation for the Tru64 UNIX operating system and related products at the following Web site:

http://www.businesslink.digital.com/

If you need help deciding which documentation best meets your needs, see the Tru64 UNIX *Documentation Overview* or call **800-344-4825** in the United States and Canada. In Puerto Rico, call **787-781-0505**. In other countries, contact your local Compaq subsidiary.

If you have access to Compaq's intranet, you can place an order at the following Web site:

http://asmorder.nqo.dec.com/

The following table provides the order numbers for the Tru64 UNIX operating system documentation kits. For additional information about ordering this and related documentation, see the *Documentation Overview* or contact Compaq.

Name	Order Number
Tru64 UNIX Documentation CD-ROM	QA-6ADAA-G8
Tru64 UNIX Documentation Kit	QA-6ADAA-GZ
End User Documentation Kit	QA-6ADAB-GZ
Startup Documentation Kit	QA-6ADAC-GZ
General User Documentation Kit	QA-6ADAD-GZ
System and Network Management Documentation Kit	QA-6ADAE-GZ
Developer's Documentation Kit	QA-6ADAG-GZ
Reference Pages Documentation Kit	QA-6ADAF-GZ

Reader's Comments

Tru64 UNIX
System Configuration and Tuning
AA-RH9GA-TE

Compaq welcomes your comments and suggestions on this manual. Your input will help us to write documentation that meets your needs. Please send your suggestions using one of the following methods:

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

If you are not using this form, please be sure you include the name of the document, the page number, and the product name and version.

Please rate this manual:

	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Usability (ability to access information quickly)	☐	☐	☐	☐

Please list errors you have found in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____
_____
_____
_____

What version of the software described by this manual are you using? _____

Name, title, department _____
Mailing address _____
Electronic mail _____
Telephone _____
Date _____

Do Not Cut or Tear - Fold Here and Tape

COMPAQ

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPAQ COMPUTER CORPORATION
UBPG PUBLICATIONS MANAGER
ZKO3-3/Y32
110 SPIT BROOK RD
NASHUA NH 03062-2698

Do Not Cut or Tear - Fold Here

Cut on This Line