DIGITAL UNIX

System Configuration and Tuning

Part Number: AA-Q0R3F-TE

December 1997

Product Version: DIGITAL UNIX Version 4.0D

This manual describes how to set up and tune high-performance and high-availability systems running the DIGITAL UNIX operating system.

Digital Equipment Corporation
Maynard, Massachusetts

Contents

About This Manual

This manual contains information about configuring systems for high performance and high availability. It describes how to determine the needs of your environment, including performance and availability requirements, and how to configure a system that will meet your current and future needs. This manual also describes how to tune systems to improve performance.

For DIGITAL UNIX Version 4.0D and higher, DIGITAL recommends that you use the graphical user interface (GUI) to 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. If your system is a workstation or a server with the CDE software, the SysMan applications are available in the Application Manager. You can access the Application Manager from the CDE Front Panel by clicking on its icon. The SysMan applications are organized into five groups within the System_Admin group. Double click on the System_Admin group to access the SysMan Configuration Checklist, the Welcome to SysMan online help volume, and the five application groups. See the *System Administration* manual for more information about accessing SysMan.

Audience

This manual is intended for system administrators who are responsible for managing a DIGITAL UNIX operating system. Administrators should have an in-depth knowledge of operating system concepts, commands, and utilities. It is also important for administrators to understand how their systems are being used. Such an understanding is crucial to successfully tuning a system for better performance.

Organization

This manual consists of six chapters, two appendixes, and a glossary:

Chapter 1	Introduces the terms and concepts related to performance and availability.
Chapter 2	Describes the tools for analyzing system resource usage.
Chapter 3	Describes how to optimize applications and CPU usage.

Chapter 4	Describes how to configure and tune the memory subsystem for high performance.
Chapter 5	Describes how to configure and tune your storage subsystem for high performance.
Chapter 6	Describes how to tune your network subsystem for high performance.
Appendix A	Describes tuning guidelines for special types of systems.
Appendix B	Describes kernel subsystem attributes.
Glossary	Lists 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 DIGITAL 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 manuals also provide useful, relevant information:

- *Technical Overview*
- *Network Administration*
- *Logical Storage Manager*
- *AdvFS Guide to File Administration*
- *DIGITAL Systems & Options Catalog*

The printed version of the DIGITAL UNIX documentation set is color coded to help specific audiences quickly find the books that meet their needs. (You can order the printed documentation from DIGITAL.) This color coding is reinforced with the use of an icon on the spines of books. The following list describes this convention:

Audience	Icon	Color Code
General users	G	Blue
System and network administrators	S	Red
Programmers	P	Purple

Audience	Icon	Color Code
Device driver writers	D	Orange
Reference page users	R	Green

Some books in the documentation set 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, Glossary, and Master Index* provides information on all of the books in the DIGITAL UNIX documentation set.

Reader's Comments

DIGITAL welcomes any comments and suggestions you have on this and other DIGITAL 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:

  Digital Equipment Corporation
  UBPG Publications Manager
  ZKO3-3/Y32
  110 Spit Brook Road
  Nashua, NH 03062-9987

  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 DIGITAL UNIX that you are using.
- If known, the type of processor that is running the DIGITAL UNIX software.

The DIGITAL 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 DIGITAL technical support office. Information provided with the software media explains how to send problem reports to DIGITAL.

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 High-Performance and High-Availability Systems

Businesses want 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 system that will meet your current and future computing needs.

This chapter introduces you to the process of configuring a system and includes information about the following topics:

- Common terms related to performance and availability (Section 1.1)
- How to obtain high availability and high performance (Section 1.2 and Section 1.3)
- How to plan your configuration (Section 1.4)
- Configuration and tuning recommendations (Section 1.5)
- The steps to configure and tune systems (Section 1.6)

Later chapters provide detailed information about monitoring systems, identifying performance problems, optimizing applications and the central processing unit (CPU), and configuring and tuning the virtual memory, storage, and network subsystems.

1.1 Terminology and Concepts

This section introduces the terms and concepts that are used to describe performance and availability.

1.1.1 System Configuration

Your system **configuration** consists of a combination of hardware and software for a single system or a cluster of systems. For example, CPUs, memory boards, the operating system, and mirrored disks are parts of a configuration. To **configure** a system, you need to set up a new or modify an existing hardware or software configuration. For example, configuring the I/O subsystem can include setting up mirrored disks.

Systems can be single-CPU systems or **multiprocessor** systems, which allow two or more processors to share common physical memory. 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.

Certain types of environments, such as large database environments, require multiprocessing systems and large storage configurations to handle the workload. Very-large memory (**VLM**) systems utilize 64-bit architecture, multiprocessing, and at least 2 GB of memory. Very-large database (**VLDB**) systems are VLM systems that also use a large and complex storage configuration. The following list describes the components of a typical VLM/VLDB 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

The **virtual memory** subsystem controls the allocation of memory to processes by using a portion of physical memory, disk swap space, and various daemons and algorithms. A **page** is the smallest portion of physical memory that the system can allocate (8 KB of memory). Virtual memory operation involves **paging**, reclaiming pages so they can be reused, and **swapping**, writing a suspended process' modified (dirty) pages to swap space, which frees large amounts of memory.

After a system is configured, you may want to **tune** the system to improve performance. You can tune a system by changing the values of kernel variables in order to modify the kernel. **Kernel variables** affect the behavior and performance of the kernel, the virtual memory subsystem, the I/O subsystems, and applications. You can temporarily modify the kernel by changing the kernel variables while the system is running, or you can permanently modify the kernel by changing the values of attributes.

Use **attributes** to modify the kernel without rebuilding the kernel. In some cases, you can modify the kernel by changing **parameter** values in the system configuration file; however, you must rebuild the kernel to use the new parameter values. See Section 2.11 for information about viewing and modifying kernel variables, attributes, and parameters.

If tuning a system does not sufficiently improve performance, you may have to reconfigure your system, which can involve adding CPUs or memory, changing the storage configuration, or modifying the software application.

1.1.2 System Performance

System performance depends on an efficient utilization of system **resources**, which are the hardware and software components (CPUs, memory, networks, and disk storage) that are available to users or applications. A system must perform well under the normal **workload** exerted on the system by the applications and the users.

The system workload changes over time. You may add users or run additional applications. You may need to reconfigure your system to handle an increasing workload. **Scalability** refers to a system's ability to utilize additional resources with a predictable increase in performance, or 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**. Potential bottlenecks include the virtual memory subsystem and I/O buses. 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 system performance requires 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 system's memory requirements.

1.1.3 Disk Performance

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.

The Unified Buffer Cache (UBC) affects disk I/O performance. The UBC is allocated a portion of physical memory to cache most-recently accessed file system data. By functioning as a layer between the operating system and the storage subsystem, the UBC is able to decrease the number of disk operations.

Disk I/O performance also depends on the characteristics of the workload's I/O operations. Data transfers can be large or small and can involve reading data from a disk or writing data to a disk.

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.

In addition, data transfers 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. Raw I/O is often used by the operating system and by database application software.

Disk I/O performance also is affected by the use of redundant array of independent disks (**RAID**) technology, which can provide both high disk I/O performance and high data availability. The DIGITAL UNIX operating system provides RAID functionality by using the Logical Storage Manager (LSM) software. DIGITAL UNIX also supports hardware-based RAID products, which provide RAID functionality by using intelligent controllers, caches, and software.

There are four primary RAID levels:

- **RAID 0**—Also known as disk **striping**, RAID 0 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.
- **RAID 1**—Also known as 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.
- **RAID 3**—A type of **parity RAID**, RAID 3 divides data blocks and distributes (stripes) the data across a disk array, providing parallel access to data and increasing bandwidth. RAID 3 also provides high 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.

To address your performance and availability needs, you can combine some RAID levels (for example, you can combine RAID 1 with RAID 0 to mirror striped disks). Some hardware-based RAID products support **adaptive RAID 3/5** (also called dynamic parity RAID), which improves disk I/O performance for a wide variety of applications by dynamically adjusting, according to workload needs, between data transfer-intensive algorithms and I/O operation-intensive algorithms.

See Section 5.2.1 for more information about RAID and RAID products.

1.1.4 High Availability

High availability is the ability of a resource to withstand a hardware or software failure. Resources (for example, systems or disk data) can be made highly available by using some form of resource duplication or **redundancy**.

For example, you can make the data on a disk highly available by mirroring that disk; that is, replicating the data on a different disk. If the original disk fails, the copy is still available to users and applications. If you use parity RAID, the redundant data is stored in the parity information, which is used to regenerate data if a disk failure occurs.

In addition, you can make the network highly available by using redundant network connections. If one connection becomes unavailable, you can still use the other connection for network access. Network availability depends on the application, the network configuration, and the network protocol.

To make a system highly available, you must set up a **cluster**, which is a loosely coupled group of servers configured as cluster member systems. In a cluster, software applications are capable of running on any member system. Some applications can run on only one member system at a time; others can run on multiple systems simultaneously. Cluster member systems usually share highly available disk data, and some clusters support a high-performance interconnect that enables fast and reliable communications between members.

A cluster utilizes **failover** to ensure application and system availability. If a member system fails, all cluster-configured applications running on that system fail over to a different member system, which restarts the applications and makes them available to users.

To completely protect a configuration from failure, you must eliminate each point of failure. An example of a configuration that has no single point of failure is as follows:

- A cluster to protect against a system failure
- Two network connections to protect against a network failure
- Disks mirrored across different buses to protect against a disk, bus, or adapter failure

For increased availability, you can use multiple layers of redundancy to protect against multiple failures. See Section 1.2 for more information about availability.

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).

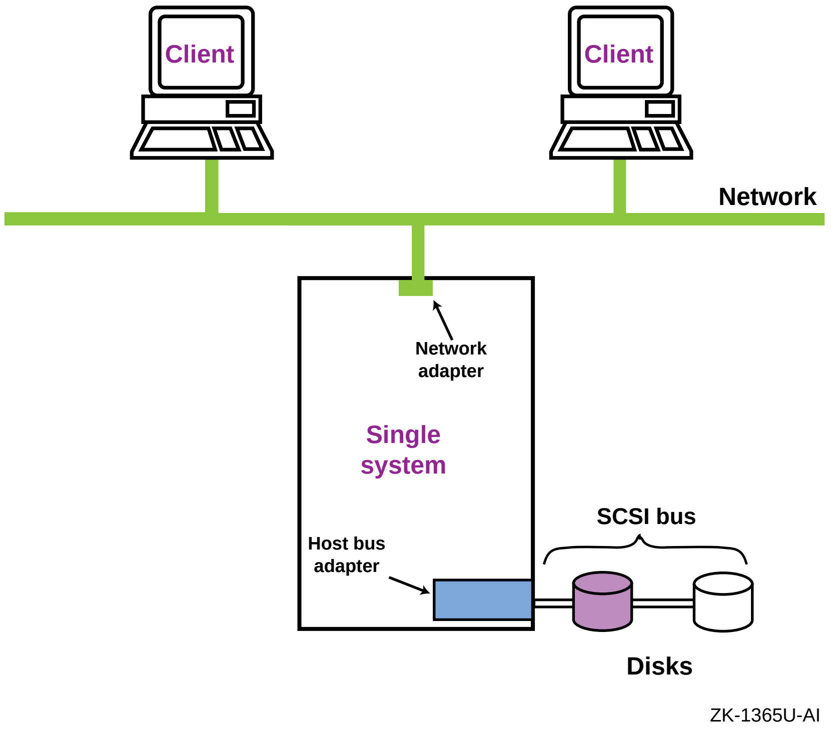
1.2 Understanding High Availability

A resource that is highly available is resistant to specific hardware and software failures. This is accomplished by duplicating resources (for example, systems, network interfaces, or data), and may also include an automatic failover mechanism that makes the resource failure virtually imperceptible to users.

There are various degrees of high availability, and you must determine how much you need for your environment. A configuration that has no single point of failure is one in which you have duplicated each vital resource. Environments that are not prone to failure or are able to accommodate down time may only require data to be highly available.

Figure 1–1 shows a configuration that is vulnerable to multiple failures, including system, network, disk, and bus failures.

Figure 1–1: Configuration With Potential Points of Failure



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

However, it is not always possible or practical to protect against every possible failure scenario or to provide multiple levels of redundancy. When planning your configuration, you must determine how much availability you need and the best way to achieve it.

Software-based RAID (LSM) and hardware-based RAID products provide you with various degrees of data availability. In addition, specific configurations can improve data availability. For example, mirroring data across buses protects against disk, bus, and adapter failures.

DIGITAL UNIX TruCluster ™ products provide high system and application availability. Brief descriptions of some cluster products are as follows:

- TruCluster Available Server Software

  Allows you to set up an available server environment (ASE), which consists of systems and disk and tape devices that are connected to shared SCSI buses. Together they provide highly available software and data to client systems. An ASE uses failover to significantly reduce down time due to hardware and software failures.

- TruCluster Production Server Software

  Provides you with high performance and highly available access to applications and data in a network environment. Production Server significantly reduces down time caused by hardware and software failures, and provides scalability beyond the limits of a single system. A Production Server configuration is similar to an ASE, but it also uses a PCI-based cluster interconnect that enables fast and reliable communications between cluster members.

The following sections describe how to eliminate points of failure, and how to increase resource availability.

1.2.1 Eliminating Points of Failure

When configuring a system for high availability, you must protect the system's resources from failure. The following list describes each potential point of failure and how to eliminate it:

- System failure

  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 hardware or software failure occurs (for example, a system crashes or an application fails). To obtain protection against a failure on a single system, you must set up a cluster with at least two member systems. If a failure occurs on one member system, the cluster-configured applications running on that system fail over to another member system, which then runs the applications.

  However, a two-member cluster is no longer a highly available configuration if one member fails, because the remaining member is now a potential point of failure. To protect against multiple system failures, you must set up a cluster with more than two member systems.

- Disk failure

  To protect against disk failure, mirror disks or use parity RAID.

- Host bus adapter or bus failure

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

- Network connection failure

  Network connections may fail because of a failed network interface or a problem in the network itself. To maintain network access if a network connection fails, install more than one network interface in a system and make sure that your applications support this functionality.

- Power failure

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

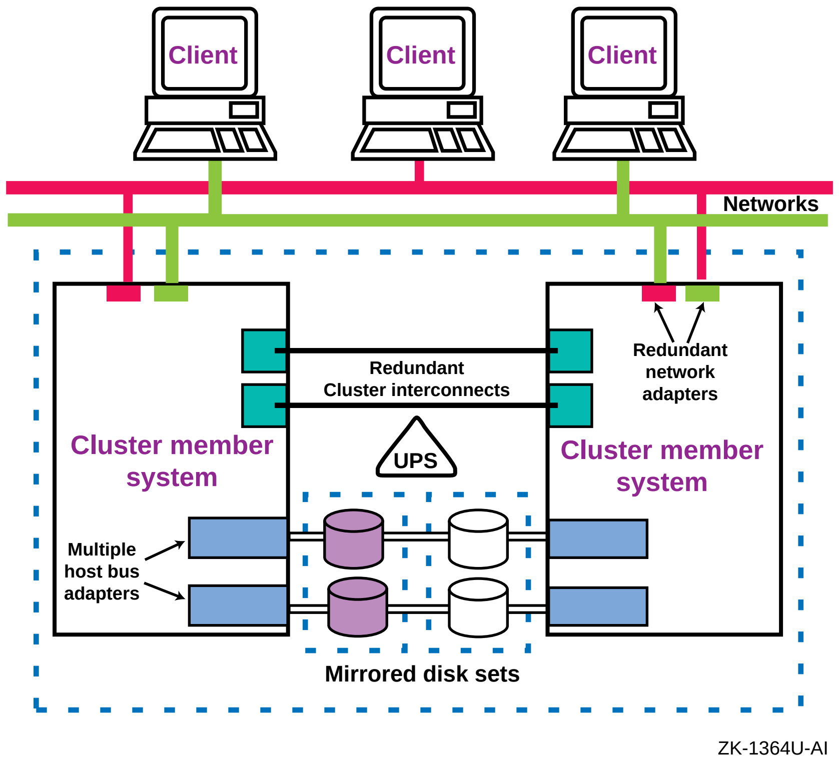
  Use an uninterruptible power system (UPS) to protect against a total power failure (for example, the power in a building fails). A UPS depends on a viable battery source and monitoring software.

- Cluster interconnect failure

  If a cluster supports high-performance cluster interconnects, you can connect each member system to redundant (two) interconnects. If one cluster interconnect fails, the cluster members can still communicate over the remaining interconnect.

Figure 1–2 shows a fully redundant cluster configuration with no single point of failure for the server systems.

Figure 1–2: Fully Redundant Cluster Configuration



Because you can never eliminate the possibility that multiple failures will make a resource or component unavailable, 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.

1.2.2 Increasing System Availability

You must decide how much system availability you need and where a system is most vulnerable to failure. Table 1–1 describes how to increase the system availability by eliminating single points of failure, as well as the tradeoffs.

Table 1–1: Increasing System Availability

To protect against:	You can:	Tradeoff:
Single system failure	Set up a cluster with at least two members	Cost of additional hardware and software, increased management complexity
	Use the lastest versions of hardware, firmware, and operating system	Possible down time during upgrade
Multiple system failures	Set up a cluster with more than two members	Cost of additional hardware and software, increased management complexity
Network connection failure	Configure multiple network connections	Cost of additional hardware, requires I/O slots
Cluster interconnect failure	Set up a second cluster interconnect	Cost of additional hardware, uses a PCI slot
Total power failure	Use a battery-backed uninterruptible power system (UPS)	Cost of UPS hardware
Cabinet power supply failure	Use redundant power supplies or mirror disks across cabinets with independent power supplies	Cost of additional hardware and decrease in write performance on mirrored disks

1.2.3 Increasing Data Availability

Not only is it important for users and applications to be able to access data easily and quickly, data needs to be available. Table 1–2 describes how to increase the availability of data by addressing points of failure, as well as the tradeoffs.

Table 1–2: Increasing Data Availability

To protect against:	You can:	Tradeoff:
Disk failure	Mirror disks	Cost of additional disks and decrease in write performance
	Use parity RAID	Cost of additional hardware and software, increase in management complexity, and performance impact under write loads
Host bus adapter or bus failure	Mirror data across disks on different buses	Cost of additional hardware and requires additional I/O bus slots
System failure	Set up a cluster	Cost of additional hardware and software, increase in management complexity
Total power failure	Use a battery-backed uninterruptible power system (UPS)	Cost of UPS hardware
Cabinet power supply failure	Use redundant power supplies or mirror disks across cabinets with independent power supplies	Cost of additional hardware and decrease in write performance on mirrored disks

1.2.4 Achieving High Availability and High Performance

Configuring a system for high availability can affect performance, depending on your configuration and the characteristics of your workload. Table 1–3 shows how high-availability solutions affect system performance.

Table 1–3: Impact of High Availability on System Performance

Availability Solution	Performance Impact
Mirroring disks	Can improve disk read performance, but may cause a degradation in write performance (you can mirror striped disks to combine the performance benefits of striping with high availability)
Mirroring disks across different buses	Prevents a single bus from becoming an I/O bottleneck
Parity RAID	Improves disk I/O performance only if all member disks are available; performance degrades as disks fail

Table 1–3: Impact of High Availability on System Performance (cont.)

Availability Solution	Performance Impact
Redundant network connections	Improves network performance and increases client access
Cluster	Improves overall performance by spreading workload across member systems, which provides applications and users with more CPU and memory resources

1.3 Understanding High Performance

A system must have a dependable level of performance to meet the needs of users and applications. You must configure your system so that it can rapidly respond to the demands of a normal workload and maintain an adequate level of performance if the workload increases.

Some environments require that a system be scalable. A scalable system allows you to add hardware (for example, CPUs) to improve performance or to absorb an increase in the workload.

You must understand the characteristics of your workload to determine the level of performance you require, and which configuration will meet your performance needs. Although some environments require the highest possible performance, this level of performance may not be necessary or cost effective.

System performance depends on the interaction between the hardware and software configuration and the workload. A system that performs well must use CPU, memory, and I/O resources efficiently. If a resource reaches its capacity, it becomes a bottleneck and can degrade performance. Bottlenecks are often interrelated; for example, insufficient memory can cause excessive paging and swapping, which may result in a bottleneck in the disk I/O subsystem.

To plan a configuration that will meet your performance needs, you must identify which resources will have the biggest impact on performance. For example, if your applications are CPU-intensive, you may want to consider a system with multiple CPUs and sufficient memory bandwidth. If the applications require a lot of memory, you must configure sufficient memory for the system. An inadequate amount of memory will degrade the overall system performance.

If your applications perform a large number of disk I/O operations, configure your storage subsystem to prevent disk and bus bottlenecks. If your system is an Internet server, you must be sure it can handle many network requests. In addition, if you require both high availability and high performance, you must determine how a high-availability configuration impacts system performance.

After you plan and set up your configuration, you may be able to improve performance by tuning the system. However, tuning may provide only marginal performance improvements, so make sure that your configuration is appropriate for your workload.

Performance problems can have various sources, including the following:

- Incorrect values for kernel variables

  Depending on your configuration and workload, you may need to modify some kernel variable values to obtain optimal performance.

- Incorrect configuration for the workload

  If tuning the system does not improve performance, your configuration may not be suitable for your workload. In addition, your resources may be inadequate for the workload. For example, you may need to increase your CPU or memory resources, upgrade to high-performance hardware, or add disks.

- Fragmented disks

  Disk fragmentation, in which file data is not contiguously located on a disk, can degrade read and write performance because multiple I/O operations are required to access a file.

- Poorly written or nonoptimized applications

  If an application is the source of a performance problem, you must rewrite or optimize the application.

The commands described in Chapter 2 can help you identify the source of a performance problem.

1.4 Planning Your Configuration

To plan your DIGITAL UNIX configuration, follow these steps:

1. Understand your workload and the characteristics of the users and applications.
2. Determine your performance and availability requirements.
3. Choose which hardware and software configuration will satisfy your performance and availability needs.

The following sections describe these steps in detail.

1.4.1 Understanding Your Workload

Before choosing a configuration to meet your needs, you must determine the impact of your workload on the system. To do this, you must understand the characteristics of your applications and users and how they utilize the software and hardware (for example, how they perform disk I/O).

Use Table 1–4 to help you understand application behavior. You may want to duplicate and fill out this table for each application.

Table 1–4: Application Characteristics

Application Name:
Describe the application objectives.
Describe the performance requirements.
Is the application CPU-intensive?
What are the application's memory needs?
How much disk storage does the application require?
Does the application require high bandwidth or throughput?
Does the application perform large sequential data transfers?
Does the application perform many small data transfers?
What is the size of the average data transfer?
What percentage of the data transfers are reads?
What percentage of the data transfers are writes?
Does the application perform many network operations?
What are your system availability requirements?
What are your data availability requirements?
What are your network availability requirements?

Use Table 1–5 to help you understand user behavior. Different users may place different demands on the system. For example, some users may be performing data processing, while others may be compiling code. You may want to duplicate and fill out this table for each type of user.

Table 1–5: User Characteristics

User Type:	
Describe the type of user.	
Specify the number of users.	
Describe the objectives of the users.	
Describe the tasks that the users perform.	
List the applications run by the users.	
What are the data storage requirements for the users?	

After you understand how your applications and users use the hardware and software, you can determine the performance and availability goals for your environment.

1.4.2 Determining Performance and Availability Goals

Before you configure a system, you must determine the goals for the environment in terms of the following criteria:

- Performance

  You must determine an acceptable level of performance for the applications and users. For example, 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

  You must determine how future expansion will affect performance. Be sure to include in your plans any potential workload increases and, if necessary, choose a configuration that is scalable or can absorb an increase in workload.

After you determine the goals for your environment, you can choose the configuration that will meet the needs of the applications and users and address your environment goals.

1.4.3 Choosing an Appropriate Configuration

After you understand the needs of your applications and users and determine your performance and availability goals, choose the hardware and software configuration that meets your needs.

You must choose a system that will provide the necessary CPU and memory resources, and that will support your network and storage configuration. Because systems have different characteristics and features, the type of system you choose determines whether you can install additional CPU or memory boards, connect multiple I/O buses, or use the system in a cluster. Systems also vary in their scalability, which will determine whether you can improve system performance by adding resources, such as CPUs.

A primary consideration for choosing a system is its CPU and memory capabilities. Some systems support multiple CPUs. Another consideration is the number of I/O bus slots in the system.

For detailed information about features for systems, network adapters, host bus adapters, RAID controllers, and disks, see the *DIGITAL Systems & Options Catalog*. For information about operating system hardware support, see the DIGITAL UNIX *Software Product Description*.

When choosing a system that will meet your needs, you must determine your requirements for the following hardware and functionality:

- Number and speed 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.

  CPUs have different processing speeds. If your environment is CPU-intensive, you may want to choose a system that supports CPUs with fast speeds. CPUs also have different sizes for on-chip caches, which provide high performance. Some systems have secondary caches that reside on the main processor board and some have tertiary caches.

See Chapter 3 for information about CPU configuration.

- Amount of memory and the number of memory boards

  You must determine the total amount of memory that you need to handle your workload. Insufficient memory resources will cause performance problems. In addition, your memory bank configuration will affect performance. You must choose a system that provides the necessary amount of memory.

  See Chapter 4 for information about memory requirements and configuration.

- Cluster support

  There are various cluster products that can provide you with high system availability. However, you can use only specific systems, adapters, controllers, and disks with the cluster products.

  In addition, some cluster products use high-performance cluster interconnects that are connected to PCI bus slots. You must ensure that a cluster system has enough PCI slots for the cluster interconnects.

  See a specific cluster product's *Software Product Description* for information about the systems and other hardware that can be used with that product.

- Number and type of network adapters

  Systems support a variety of network adapters that you use to connect to a network. Adapters have different performance features. In addition, you can use multiple network connections to improve network availability. You must choose a system that supports the network adapters that you require, and that has enough I/O slots available for the adapters.

  See Chapter 6 and the *Network Administration* manual for information about network configuration.

- Number and type of host bus adapters

  Systems use buses to communicate with devices. Host bus adapters are used to communicate between buses. Host bus adapters are installed in I/O bus slots, so you must choose a system that has enough I/O slots available for the adapters.

  See Chapter 5 for information about storage configurations.

- Number and type of RAID controllers

  You can connect only a limited number of devices to a SCSI bus. The SCSI-2 specification allows 8 devices on each SCSI bus, and the SCSI-3 specification allows 16 devices on a bus. A RAID controller allows you to increase the number of SCSI buses that can be accessed through a single I/O bus slot.

Some RAID controllers are installed directly in I/O bus slots, while others are connected to systems through a host bus adapter installed in an I/O bus slot. You must choose a system that supports RAID controllers and has a sufficient number of I/O bus slots available for the controllers.

See Chapter 5 for information about hardware RAID configurations.

Table 1–6 can help you identify the characteristics of a system that will meet your needs.

Table 1–6: System Characteristics

If you require:	You need a system that:
Multiprocessing support	Supports multiprocessing and the number of CPUs that you want.
Fast processing time	Supports CPUs with fast speeds and fast memory.
Additional memory boards	Has backplane slots available for memory boards.
Cluster support	Supports the cluster product that you want to use.
Network adapters	Supports the network adapters that you want to use, and has an I/O slot available for each adapter.
Host bus adapters	Supports the host bus adapters that you want to use, and has an I/O slot available for each adapter.
RAID controllers	Supports the RAID controllers that you want to use, and has an I/O bus slot available for each controller.
Cluster interconnects	Has a PCI slot available for each interconnect.

Fill in the requirements listed in Table 1–7 to get a profile of the system that will meet your needs.

Table 1–7: System Requirements

Feature:	Requirement:
Number of CPU boards:	
CPU processing speed:	
Total amount of memory:	
Number of memory boards:	
Cluster support:	

Table 1–7: System Requirements (cont.)

Feature:	Requirement:
Type and number of network adapters:	
Type and number of host bus adapters:	
Type and number of backplane RAID controllers:	
Number of cluster interconnects:	

1.5 Primary Configuration and Tuning Recommendations

This manual describes many configuration and tuning tasks that you can use to improve system performance. Some of the recommendations can greatly improve performance. However, many of the recommendations provide only marginal improvement and should be used with caution.

To help you configure and tune your system, there are recommendations to follow that will provide you with the best performance improvement for most configurations. Many of these recommendations are used by the `sys_check` utility, which gathers performance information and outputs this information in an easy-to-read format. The `sys_check` utility uses some of the tools described in Chapter 2 to check your configuration and kernel variable settings and provides warnings and tuning recommendations if necessary.

To obtain the `sys_check` utility, access the following location or call your customer service representative:

```
ftp://ftp.digital.com/pub/DEC/IAS/sys_check
```

The following list describes the primary tuning recommendations. If these recommendations do not solve your performance problem, use the other recommendations described in this manual.

- Operating system and kernel recommendations
  - Ensure that you are using the latest patches for the operating system. Examine the system startup messages or use the DECevent utility to show the operating system revision.
  - Ensure that you are using the latest firmware for your system, adapters, controllers, and disks. Examine the system startup messages or use the DECevent utility to show firmware revisions.

  - Ensure that important applications have high priority. Use the `nice` command or the Class Scheduler to assign CPU priorities. See Chapter 3.
  - Apply any kernel variable modifications that are recommended for your type of configuration (for example, an Internet server). See Appendix A.
- Memory recommendations
  - Ensure that you have sufficient memory for your configuration. See Section 4.6.1.
  - Ensure that your system has sufficient swap space and distribute swap space across different disks and buses. See Section 4.6.2.
  - Increase the address space available to processes. See Section 4.7.3.
  - Increase the system resources available to processes. See Section 4.7.4, Section 4.7.5, Section 4.7.6, Section 4.7.7, Section 4.7.8 and Section 5.3.3.1.
  - Reduce application memory requirements. See Section 4.7.10.
  - If your system does few disk I/O operations, reduce the amount of memory allocated to the Unified Buffer Cache (UBC). See Section 4.8.
  - Modify the rate of swapping. See Section 4.7.12.
  - Modify the rate of dirty page prewriting. See Section 4.7.13.
- General disk and I/O recommendations
  - Use high-performance hardware. See Section 5.3.1.
  - Distribute disk I/O and file systems across different disks and multiple buses. See Section 5.3.2.2 and and Section 5.3.2.3. You can distribute disk and file system I/O by striping data across multiple disks. See Section 5.2.1.
  - Defragment file systems. See Section 5.6.2.2 and Section 5.7.2.1.
  - If your applications are disk I/O-intensive, increase the amount of memory allocated to the UBC. See Section 5.3.3.4.
  - Increase the maximum number of open files. See Section 5.3.3.1.
  - Increase the size of the namei cache. See Section 5.3.3.2.
- Advanced File System (AdvFS) recommendations
  - Use multiple-volume file domains. See Section 5.6.1.1.
  - Increase the amount of memory allocated to the AdvFS buffer cache. See Section 5.6.2.1.

  - Increase the dirty data caching threshold. See Section 5.6.2.3.
  - Decrease the I/O transfer read-ahead size. See Section 5.6.2.4.
  - Disable the flushing of dirty pages mapped with the `mmap` function during a `sync` call. See Section 5.6.2.5.
  - Modify the AdvFS device queue limit. See Section 5.6.2.6.
- UNIX File System (UFS) recommendations
  - Modify the file system fragment size. See Section 5.7.1.1.
  - Increase the size of metadata buffer cache. See Section 4.9.1.
  - Delay flushing full write buffers to disk. See Section 5.7.2.2.
- Network recommendations (Internet servers)
  - Increase the size of the hash table that the kernel uses to look up TCP control blocks. See Section 6.1.1.
  - Increase the limits for partial TCP connections on the socket listen queue. See Section 6.1.2.
  - Increase the maximum number of concurrent nonreserved, dynamically allocated ports. See Section 6.1.3.
  - Enable TCP keepalive functionality. See Section 6.1.4.
- Network File System (NFS) recommendations
  - Ensure that you have a sufficient number of `nfsd` daemons running on the server. See Section 6.2.2.
  - Ensure that you have a sufficient number of `nfsiod` daemons running on the client. See Section 6.2.3.

1.6 Steps to Configure and Tune Systems

Setting up and maintaining a high-performance or high-availability system requires a number of steps. The process is as follows:

1. Configure the system.

   To configure (or reconfigure) a system, you must determine the requirements of your environment and choose a configuration to meet your needs. Then, you can set up the hardware, operating system, layered products, and applications.

2. Perform any recommended initial tuning tasks.

   For some configurations, you may have to perform some tuning tasks immediately after you configure your system. For example, if your system is used as an Internet server, follow the recommendations to modify the default values of system parameters and attributes.

3. Monitor system performance.

   You must carefully monitor the performance of your system, as described in Chapter 2.

   If system performance is acceptable, you must continue to monitor the system on a consistent basis, because performance may degrade if resources reach their capacity or if there is a significant change in the environment (for example, you increase the workload or you reconfigure the system).

   If system performance is not acceptable, you must determine the source of the problem.

4. Identify the source of the performance problem.

   Use the tools described in Chapter 2 to locate the source of the problem. The *DIGITAL Systems & Options Catalog* contains information about the capacity of hardware resources.

5. Determine if there is a tuning solution that will eliminate the performance problem.

   If there is no tuning solution or if you have exhausted all possible tuning solutions, you may have to reconfigure the system to eliminate the performance problem.

6. Eliminate the performance problem.

   To eliminate a performance problem, first try simple, no-cost solutions, such as running applications at offpeak hours or restricting disk access. Then, you can try more complex and expensive solutions, such as tuning the system or adding more hardware. Section 1.5 includes a list of the primary tuning tasks that may help you to improve performance.

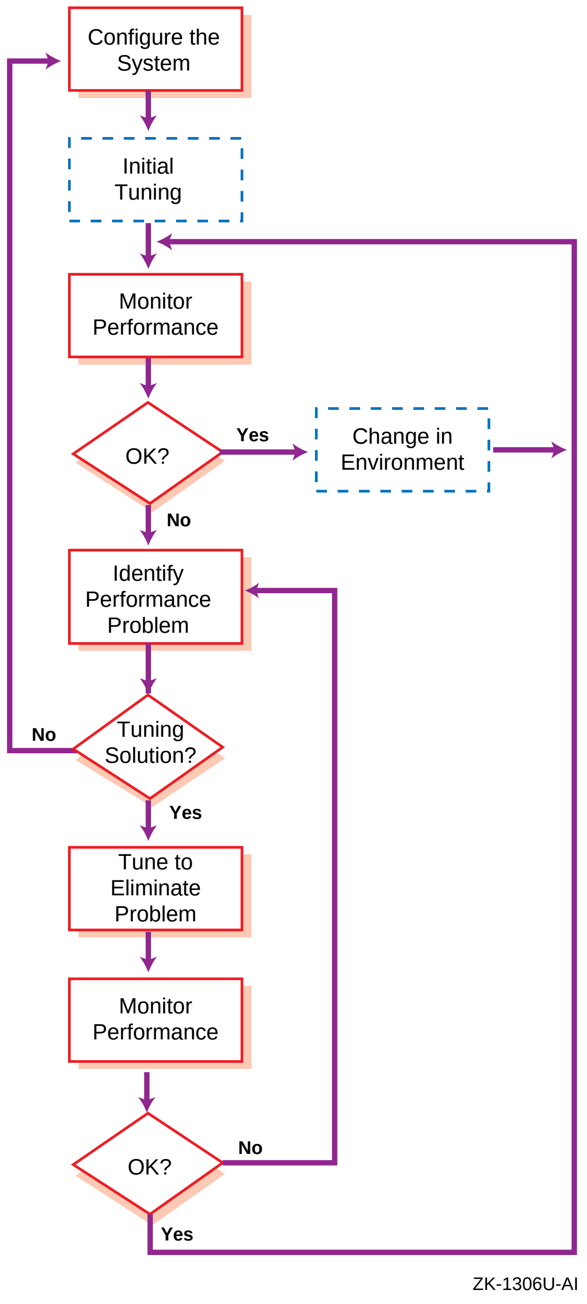
   If you are sure your CPU and applications are optimized, tuning the virtual memory subsystem provides the best performance benefit and should be the primary area of focus. If tuning memory does not eliminate the problem, tune the I/O subsystem. Tuning usually requires modifying kernel attributes. However, you may be able to improve system performance by performing some administrative tasks, such as defragmenting file systems or modifying stripe widths.

7. Monitor system performance.

   After you tune the system, you must carefully monitor the system to ensure that the performance problem has been eliminated. If a tuning recommendation does not eliminate the problem, try another recommendation. If you cannot reduce or eliminate a performance problem by tuning the system, you must reconfigure the system.

The flowchart shown in Figure 1–3 describes the configuration and tuning process. Detailed information about diagnosing performance problems and information about configuring and tuning the CPU, virtual memory, storage, and networks is discussed in later chapters.

Figure 1–3: Configuration and Tuning Process



2 Diagnosing Performance Problems

To get the maximum performance from a system, you must eliminate any performance bottlenecks. Diagnosing performance problems involves identifying the problem (for example, excessive paging and swapping), and then determining the source of the problem (for example, insufficient memory or incorrect virtual memory subsystem attribute values).

This chapter describes how to gather and analyze information that will help you diagnose performance problems. This chapter also describes how to modify kernel variables, attributes, and parameters. Later chapters describe how to correct performance problems found in various subsystems.

2.1 Checking System Performance

Although performance problems often are readily apparent (for example, applications complete slowly or the system logs messages stating that it is out of resources), other problems may not be obvious to users or administrators. In addition, it may be difficult to identify the source of the problem.

There are several ways to determine if a system has a performance problem or if you can improve system performance. Some indications of performance problems are as follows:

- Slow system response time

  If users complain about a slow system response time, this may indicate that processes are being swapped out because of a lack of memory. The `ps` command displays information about swapped-out processes. See Section 2.4.1 for more information.

- Slow application completion time

  If the application completion time is inadequate, this may indicate inadequate memory or CPU power, an I/O bottleneck, or a poorly designed application. The `ps` command displays information about an application's use of system resources. See Section 2.4.1 for more information.

  Use process accounting commands to obtain information about a process' use of memory, CPU, and I/O resources and its completion time. See `accton`(8) for more information.

Use the profiling and debugging commands to analyze applications. See Table 2–7 for more information.

- Unbalanced use of disks

  Excessive activity on only a few disks may indicate an uneven distribution of disk I/O. Use the `iostat` command to display the utilization of disks. Use the `swapon -s` command to display the utilization of swap disk space. Use the `volstat` command to display information about the LSM I/O workload. See Section 2.4.4, Section 2.5.1, and Section 2.8.2 for more information.

- Excessive paging and swapping

  A high rate of paging and swapping may indicate inadequate memory for the workload. Use the `vmstat` command to display information about paging and memory consumption. See Section 2.4.2 for more information.

- Slow or incomplete network connections

  Prematurely dropped network connections may be the cause of network performance problems. Use the `netstat` command to display information about dropped network connections. See Section 2.9.1 for more information. Use the `ping` command to determine if a remote system is available. See `ping`(8) for more information.

- Event messages indicating problems in the system

  The information in the ASCII system log files or the binary log files may alert you to potential or existing performance problems. Use the DECevent utility, the `dia`, or the `uerf` command to examine the binary log files. See Section 2.2.1 for more information.

- Information from kernel and program analysis tools

  The output of profiling and debugging tools that are used to analyze code may indicate areas of code that are degrading performance or using excessive resources. See Table 2–7 for more information.

The following sections describe how to obtain information that will help you identify a performance problem and its source.

2.2 Obtaining Performance Information

To determine how your system is performing and to help diagnose performance problems, you must obtain information about your system. To do this, you need to log system events and monitor resources.

In addition, you must gather performance statistics under different conditions. For example, gather information when the system is running well and when system performance is poor. This will allow you to compare different sets of data.

After you set up your environment, immediately start to gather performance information by performing the following tasks:

- Configure event logging (Section 2.2.1)
- Set up system accounting and disk quotas to track resource utilization by each user (Section 2.2.2)
- Set up a routine to continuously monitor performance (Section 2.2.3)
- Gather initial performance statistics (Section 2.2.4)

The following sections describe these tasks in detail.

2.2.1 Configuring Event Logging

The DIGITAL UNIX operating system uses the system event-logging facility and the binary event-logging facility to log system events. The log files can help you diagnose performance problems.

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. The daemon then writes the messages to a local file or forwards the messages to a remote system, as specified in the `/etc/syslog.conf` default event-logging configuration file. See `syslogd`(8) for more 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 DECevent utility, which translates the records from binary format to ASCII format. DECevent features can analyze the information and isolate the cause of the error. DECevent also can continuously monitor the log file and display information about system events.

You must register a license Product Authorization Key (PAK) to use DECevent's analysis and notification features, or these features may also be available as part of your DIGITAL service agreement. A PAK is not needed to use DECevent to translate the binary log file to ASCII format. See Section 2.2.3.1 for more information about DECevent.

You can also use the `dia` or the `uerf` command to translate binary log files to ASCII format. See `dia`(8) and `uerf`(8) for information.

After you install the operating system, you can customize system and binary event logging by modifying the default configuration files. See the *System Administration* manual and the *Release Notes* for more information about configuring event logging.

2.2.2 Setting up System Accounting and Disk Quotas

System accounting allows you to obtain information about how users utilize resources. You can obtain information about the amount of CPU time and connect time, the number of processes spawned, memory and disk usage, the number of I/O operations, and the number of printing operations.

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 setting up system accounting and UNIX file system (UFS) disk quotas. See the Advanced File System (AdvFS) documentation for information about AdvFS quotas.

2.2.3 Choosing How to Monitor System Events

DIGITAL recommends that you set up a routine to continuously monitor system performance and to alert you when serious problems occur. There are a number of products and commands that provide system monitoring:

- DECevent utility

  Monitors system events through the binary event-logging facility, analyzes events, and performs event notification. See Section 2.2.3.1 for more information.

- Performance Manager

  Simultaneously monitors multiple DIGITAL UNIX systems, detects performance problems, and performs event notification. See Section 2.2.3.2 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 2.2.3.3 for more 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.

- `volwatch`

  Monitors the Logical Storage Manager (LSM) for failures in disks, volumes, and plexes, and sends mail if a failure occurs. See Section 2.8.4 for information.

- `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, and can also reset the statistics information to zero. See Section 2.8.2 for information.

The following sections describe the DECevent utility, Performance Manager, and Performance Visualizer in detail.

2.2.3.1 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 PAK to use the DECevent analysis and notification features.

2.2.3.2 Using Performance Manager

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

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, data, and displays you want to monitor. You can customize and extend PM, so you can 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 general system performance. The PM analyses include suggestions for eliminating problems. PM automatically discovers cluster members when a single cluster member node is specified, and it can monitor both individual cluster members and an entire cluster concurrently.

See the Performance Manager online documentation for more information.

2.2.3.3 Using Performance Visualizer

Performance Visualizer is a valuable tool for developers of parallel applications. Because it monitors 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 effects of these changes. Performance Visualizer is a DIGITAL UNIX layered product and requires a license.

Performance Visualizer also helps you identify overloaded systems, underutilized resources, active users, and busy processes.

Using Performance Visualizer, 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.

2.2.4 Gathering Performance Statistics

Use the commands described in this chapter to gather performance statistics to benchmark your system, and to help identify performance problems. It is important to gather statistics from a variety of conditions. For example, gather information at the following opportunities:

- Immediately after you install the system and before any applications are running to obtain baseline system performance information
- When the system is running well under a normal workload
- When the system has poor performance under a normal workload
- After you have tuned or reconfigured the system

In addition, you may want to use the `sys_check` utility to check your configuration and kernel variable settings. The `sys_check` utility uses some of the tools described in Section 2.3 to gather performance information and outputs this information in an easy-to-read format. The `sys_check` utility provides warnings and tuning recommendations if necessary. To obtain the `sys_check` utility, access the following location or call your customer service representative:

```
ftp://ftp.digital.com/pub/DEC/IAS/sys_check
```

See Section 2.3 for a list of tools that you can use to gather information about your system.

2.3 Performance Monitoring Tools Overview

There are various utilities and commands that you can use to gather performance statistics and other information about the system. You may have to use a combination of tools to obtain a comprehensive picture of your system.

It is important for you to gather information about your system while it is running well, in addition to when it has poor performance. Comparing the two sets of data will help you to diagnose performance problems.

In addition to tools that gather system statistics, there are application profiling tools that allow you to collect statistics on CPU usage, call counts, call cost, memory usage, and I/O operations at various levels (for example, at a procedure level or at an instruction level). Profiling allows you to identify sections of code that consume large portions of execution time. In a typical program, most execution time is spent in relatively few sections of code. To improve performance, the greatest gains result from improving coding efficiency in time-intensive sections. There also are tools that you can use to debug or profile the system kernel and collect CPU statistics and other information.

The following tables describe the tools that you can use to gather resource statistics and profiling information. In addition, there are many freeware programs available in prebuilt formats on the DIGITAL UNIX Freeware CD-ROM. These include the `top`, `lsof`, and `monitor` commands. You can also use the Continuous Profiling Infrastructure `dcpi` tool, which provides continuous, low-overhead system profiling. The `dcpi` tool is available from the DIGITAL Systems Research Center at the following location:

`http://www.research.digital.com/SRC/dcpi`

Table 2–1 describes the tools you can use to gather information about CPU and memory usage.

Table 2–1: CPU and Memory Monitoring Tools

Name	Use	Description
`vmstat`	Displays virtual memory and CPU usage statistics (Section 2.4.2)	Displays information about process threads, virtual memory usage (page lists, page faults, pageins, and pageouts), 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 2.4.1)	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 pageins; the percentage of real memory (resident set) usage; the resident set size; and the virtual address size.

Table 2–1: CPU and Memory 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 2.4.4)	Displays the total amount of allocated swap space, swap space in use, and free swap space, and also displays this information for each swap device. You can also use the `swapon` command to allocate additional swap space.
`uptime`	Displays the system load average (Section 2.4.3)	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.
`memx`	Exercises system memory	Exercises memory by running a number of processes. You can specify the amount of memory to exercise, the number of processes to run, and a file for diagnostic output. Errors are written to a log file. See `memx`(8) for more information.

Table 2–1: CPU and Memory Monitoring Tools (cont.)

Name	Use	Description
shmx	Exercises shared memory	Exercises shared memory segments by running a shmxb process. The shmx and shmxb processes alternate writing and reading the other process' data in the shared memory segments. You can specify the number of memory segments to test, the size of the segment, and a file for diagnostic output. Errors are written to a log file. See shmx(8) for more information.
kdbx cpustat	Reports CPU statistics (Section 2.4.5)	Displays CPU statistics, including the percentages of time the CPU spends in various states.
kdbx lockstats	Reports lock statistics (Section 2.4.6)	Displays lock statistics for each lock class on each CPU in the system.
dbx print vm_perfsum	Reports virtual memory statistics (Section 2.4.7)	You can check virtual memory by using the dbx debugger and examining the vm_perfsum data structure, which contains information about page faults, swap space, and the free page list.

Table 2–2 describes the tools you can use to obtain information about disk activity and usage.

Table 2–2: General Disk Monitoring Tools

Name	Use	Description
iostat	Displays disk and CPU usage (Section 2.5.1)	Displays transfer statistics for each disk, and the percentage of time the system has spent in user mode, in user mode running low priority (nice) processes, in system mode, and in idle mode.
diskx	Tests disk driver functionality	Reads and writes data to disk partitions. The diskx exerciser analyzes data transfer performance, verifies the disktab database file entry, and tests reads, writes, and seeks. The diskx exerciser can destroy the contents of a partition. See diskx(8) for more information.
dbx print nchstats	Reports namei cache statistics (Section 2.5.2)	Reports namei cache statistics, including hit rates.

Table 2–2: General Disk Monitoring Tools (cont.)

Name	Use	Description
`dbx print vm_perfsum`	Reports UBC statistics (Section 2.5.3)	Reports Unified Buffer Cache (UBC) statistics, including the number of pages of memory that the UBC is using.
`dbx print xpt_qhead`, `ccmn_bp_head`, and `xpt_cb_queue`	Reports Common Access Method (CAM) statistics (Section 2.5.4)	Reports CAM statistics, including information about buffers and completed I/O operations.

Table 2–3 describes the tools you can use to obtain information about the UNIX File System (UFS).

Table 2–3: UFS Monitoring Tools

Name	Use	Description
`dumpfs`	Displays UFS information (Section 2.6.1)	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 2.6.2)	Reports statistics on how the system is performing cluster read and write transfers.
`dbx print bio_stats`	Reports UFS metadata buffer cache statistics (Section 2.6.3)	Reports statistics on the metadata buffer cache, including superblocks, inodes, indirect blocks, directory blocks, and cylinder group summaries.
`fsx`	Exercises file systems	Exercises UFS and AdvFS file systems by creating, opening, writing, reading, validating, closing, and unlinking a test file. Errors are written to a log file. See `fsx`(8) for more information.

Table 2–4 describes the tools you can use to obtain information about the Advanced File System (AdvFS).

Table 2–4: AdvFS Monitoring Tools

Name	Use	Description
`advfsstat`	Displays AdvFS performance statistics (Section 2.7.1)	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.
`advscan`	Identifies disks in a file domain (Section 2.7.2)	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 2.7.3)	Allows you to determine if files are 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 2.7.4)	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 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.

Table 2–4: AdvFS Monitoring Tools (cont.)

Name	Use	Description
`showfsets`	Displays AdvFS fileset information for a file domain (Section 2.7.5)	Displays information about the filesets in a file domain, including the fileset names, the total number of files, the number of free 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.
`fsx`	Exercises file systems	Exercises AdvFS and UFS file systems by creating, opening, writing, reading, validating, closing, and unlinking a test file. Errors are written to a log file. See `fsx`(8) for more information.

Table 2–5 describes the commands you can use to obtain information about the Logical Storage Manager (LSM).

Table 2–5: LSM I/O Performance and Event Monitoring Tools

Name	Use	Description
`volprint`	Displays LSM disk configuration information (Section 2.8.1)	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.
`volstat`	Displays LSM I/O performance statistics (Section 2.8.2)	Displays performance statistics since boot time for all LSM objects (volumes, plexes, subdisks, and disks). 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 in a specified interval of time. The `volstat` utility also can reset the I/O statistics. See `volstat`(8) for more information.
`voltrace`	Tracks I/O operations on LSM volumes (Section 2.8.3)	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.

Table 2–5: LSM I/O Performance and Event Monitoring Tools (cont.)

Name	Use	Description
`volwatch`	Monitors LSM for object failures (Section 2.8.4)	Monitors LSM for failures in disks, volumes, and plexes, and sends mail if a failure occurs. The `volwatch` script starts automatically when you install LSM. See `volwatch`(8) for more information.
`dxlsm`	Displays statistics on LSM objects (Section 2.8.5)	Using the Analyze menu, displays information about LSM disks, volumes, and subdisks. See `dxlsm`(8) for more information.

Table 2–6 describes the commands you can use to obtain information about network operations.

Table 2–6: Network Monitoring Tools

Name	Use	Description
`netstat`	Displays network statistics (Section 2.9.1)	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.
`nfsstat`	Displays network and NFS statistics (Section 2.9.2)	Displays Network File System (NFS) and Remote Procedure Call (RPC) statistics for clients and servers, including the number of packets that had to be retransmitted (`retrans`) and the number of times a reply transaction ID did not match the request transaction ID (`badxid`).

Table 2–6: Network Monitoring Tools (cont.)

Name	Use	Description
`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.
`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 in order 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.
`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. The number and percentage of packets received in each category appears on the screen in a continuously updated display. Your kernel must be configured with the `packetfilter` option to use the command. See `nfswatch`(8) and `packetfilter`(7) for more information.

Table 2–6: Network Monitoring Tools (cont.)

Name	Use	Description
`sobacklog_hiwat` attribute	Reports the maximum number of pending requests to any server socket (Section 2.9.3)	Allows you to display the maximum number of pending requests to any server socket in the system.
`sobacklog_drops` attribute	Reports the number of backlog drops that exceed a socket's backlog limit (Section 2.9.3)	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's backlog limit.
`somaxconn_drops` attribute	Reports the number of drops that exceed the value of the `somaxconn` attribute (Section 2.9.3)	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).
`ps axlmp`	Displays information about idle threads (Section 2.9.4)	Displays information about idle threads on a client system.

Table 2–7 describes the commands you can use to obtain information about the kernel and applications. Detailed information about these profiling and debugging tools is located in the *Programmer's Guide* and the *Kernel Debugging* manual.

Table 2–7: 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* manual 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`(5) for more information.
`hiprof`	Produces a profile of procedure execution times in an application	An `atom`-based program profiling tool that produces a flat profile, which shows the execution time spent in any given procedure, and a hierarchical profile, which shows the time spent in a given procedure and all of its descendents. The `hiprof` tool uses code instrumentation rather than PC sampling to gather statistics. The `gprof` command is usually used to filter and merge output files and to format profile reports. See `hiprof`(5) for more information.
`pixie`	Profiles basic blocks in an application	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`(5) for more information.

Table 2–7: 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 PC 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.
`kprofile`	Produces a PC 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.
`uprofile`	Profiles user code in an application	Profiles user code using performance counters in the Alpha chip. The `uprofile` tool allows you to profile only the executable part of a program. The `uprofile` tool does not collect information on shared libraries. You process the performance data collected by the tool with the `prof` command. See the *Kernel Debugging* manual or `uprofile`(1) for more information.

Table 2–7: 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 tailored specifically to debugging kernel code and displays 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 `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.

2.4 Gathering CPU and Virtual Memory Information

Use the following commands to obtain information about CPUs and the virtual memory subsystem:

- The `ps` command displays CPU and memory usage by processes. See Section 2.4.1.
- The `vmstat` command displays virtual memory and CPU statistics. See Section 2.4.2.

- the `uptime` command displays the system load average. See Section 2.4.3.
- The `swapon` command displays information about swap space utilization. See Section 2.4.4.
- The `kdbx cpustat` extension reports CPU statistics. See Section 2.4.5.
- The `kdbx lockstats` extension reports lock statistics. See Section 2.4.6.
- The `dbx vm_perfsum` data structure reports virtual memory information. See Section 2.4.7.

The following sections describe these commands in detail.

In addition, you can use the following commands to obtain CPU and virtual memory information:

- The `ipcs` command displays IPC statistics. See `ipcs`(1).
- The `w` command reports system load averages and user information. See `w`(1).
- The `xload` command monitors the system load. See `xload`(1X).
- The `memx` exerciser tests system memory. See `memx`(8).
- The `shmx` exerciser tests shared memory. See `shmx`(8).

2.4.1 Using ps to Display CPU and Memory Usage

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. Note that 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:

```
# 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 virtual memory allocated to the process.
4. Real memory (resident set) size of the process (`RSS`)—This is the total amount of physical memory 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)
   - Uninterruptible sleeping (U)
   - Sleeping (S)
   - Idle (I)
   - Stopped (T)
   - Halted (H)
   - Swapped out (W)
   - 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`).
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 and whether processes are swapped out. Concentrate on processes that are runnable 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 a problem with memory usage.
- 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 lower the process' priority by using either the `nice` or `renice` command. These commands have no effect on memory usage by a process.

- 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 virtual memory problem.

For information about memory tuning, see Chapter 4. For information about improving the performance of your applications, see the *Programmer's Guide*.

2.4.2 Using vmstat to Display Virtual Memory and CPU Statistics

The `vmstat` command shows the virtual memory, process, and total 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. Because the CPU operates faster than the rest of the system, performance bottlenecks usually exist in the memory or I/O subsystems.

To determine the amount of memory on your system, use the `uerf -r 300` command. The beginning of the listing shows the total amount of physical memory (including wired memory) and the amount of available memory.

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

```
# 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
```

The `vmstat` command includes information that you can use to diagnose CPU and virtual memory problems. The following fields are particularly important:

1 Virtual memory information (`memory`), including the number of pages that are on the active list, including inactive pages and Unified Buffer Cache least-recently used (UBC LRU) pages (`act`); the number of pages on the free list (`free`), and the number of pages on the wire list (`wire`). Pages on the wire list cannot be reclaimed. See Chapter 4 for more information on page lists.

2 The number of pages that have been paged out (pout).

3 Interrupt information (intr), including the number of nonclock device interrupts per second (in), the number of system calls called per second (sy), and the number of task and thread context switches per second (cs).

4 CPU usage information (cpu), including the percentage of user time for normal and priority processes (us), the percentage of system time (sy), and the percentage of idle time (id). User time includes the time the CPU spent executing library routines. System time includes the time the CPU spent executing system calls.

When diagnosing a bottleneck situation, keep the following issues in mind:

- Invoke the vmstat command when the system is idle and also when the system is busy in order to compare the resulting data. You can use the memx memory exerciser to put a load on the memory subsystem.
- Is the system workload normal? Ensure that poor performance is not caused by an atypical event or by a temporary increase in resource demand.
- Check the size of the free page list (free). Compare the number of free pages to the values for the active pages (act) and the wired pages (wire). The sum of the free, active, and wired pages should be close to the amount of physical memory in your system. Although the value for free should be small, if the value is consistently small (less than 128 pages) and accompanied by excessive paging and swapping, you may have a physical memory problem.
- Examine the pout field. If the number of pageouts is consistently high, you may have insufficient memory. You also may have insufficient swap space or your swap space may be configured inefficiently. Use the swapon -s command to display your swap device configuration, and use the iostat command to determine which swap disk is being used the most.
- 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 presents information on how to interpret the values for user, idle, and system time:

  – A high user time and a low 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.

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

  A high 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 be doing a lot of paging and swapping I/O, and therefore may have a virtual memory performance problem.

– If the idle time is very low but performance is acceptable, your system is utilizing its CPU efficiently.

  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, one or more kernel data structures is being exhausted, or you may have a hardware or kernel resource problem such as an application, disk I/O, or network bottleneck.

See Chapter 3 for information on improving CPU performance and Chapter 4 for information on tuning memory.

2.4.3 Using uptime to Display the Load Average

The `uptime` command shows how long a system has been running and the load average. The load average counts jobs that are waiting for disk I/O, and applications whose priorities have been changed with either the `nice` or `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:

```
# 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. You can also lower the priority of applications with the `nice` or `renice` command to conserve CPU cycles.

2.4.4 Using swapon to Display Swap Space Usage

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:

```
# swapon -s
Swap partition /dev/rz2b (default swap):
    Allocated space:        16384 pages (128MB)
    In-use space:               1 pages (  0%)
    Free space:             16383 pages ( 99%)

Swap partition /dev/rz12c:
    Allocated space:       128178 pages (1001MB)
    In-use space:               1 pages (  0%)
    Free space:            128177 pages ( 99%)


Total swap allocation:
    Allocated space:       144562 pages (1129MB)
    Reserved space:          2946 pages (  2%)
    In-use space:               2 pages (  0%)
    Available space:       141616 pages ( 97%)
```

See Chapter 4 and Chapter 5 for information on how to configure swap space. Use the `iostat` command to determine which disks are being used the most.

2.4.5 Checking CPU Usage With kdbx cpustat

The `kdbx 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)

By default, `kdbx` displays statistics for all CPUs in the system.

For example:

```
(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.

2.4.6 Checking Lock Usage With kdbx lockstats

The `kdbx 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

See the *Kernel Debugging* manual and `kdbx`(8) for more information.

2.4.7 Checking Virtual Memory With dbx vm_perfsum

You can check virtual memory by using the `dbx` command and examining the `vm_perfsum` data structure.

An example of the `dbx print vm_perfsum` command is as follows:

```
(dbx) print vm_perfsum
struct {
    vpf_pagefaults = 2657316
    vpf_kpagefaults = 23527
    vpf_cowfaults = 747352
    vpf_cowsteals = 964903
    vpf_zfod = 409170
    vpf_kzfod = 23491
    vpf_pgiowrites = 6768
    vpf_pgwrites = 12646
    vpf_pgioreads = 981605
    vpf_pgreads = 80157
    vpf_swapreclaims = 0
    vpf_taskswapouts = 1404
    vpf_taskswapins = 1386
    vpf_vmpagesteal = 0
    vpf_vmpagewrites = 7304
    vpf_vmpagecleanrecs = 14898
    vpf_vplmsteal = 36
    vpf_vplmstealwins = 33
    vpf_vpseqdrain = 2
    vpf_ubchit = 3593
    vpf_ubcalloc = 133065
    vpf_ubcpushes = 3
    vpf_ubcpagepushes = 3
    vpf_ubcdirtywra = 1
    vpf_ubcreclaim = 0
    vpf_ubcpagesteal = 52092
    vpf_ubclookups = 2653080
    vpf_ubclookuphits = 2556591
    vpf_reactivate = 135376
    vpf_allocatedpages = 6528
    vpf_vmwiredpages = 456
    vpf_ubcwiredpages = 0
    vpf_mallocpages = 1064
    vpf_totalptepages = 266
    vpf_contigpages = 3
    vpf_rmwiredpages = 0
    vpf_ubcpages = 2785
    vpf_freepages = 190
    vpf_vmcleanpages = 215
    vpf_swapspace = 8943
}
(dbx)
```

Important fields include the following:

- `vpf_pagefaults`—Number of hardware page faults
- `vpf_swapspace`—Number of pages of swap space not reserved

To obtain information about the current use of memory, use the `dbx print` command to display the values of the following kernel variables:

- `vm_page_free_count`—Number of pages on the free list
- `vm_page_active_count`—Number of pages on the active list
- `vm_page_inactive_count`—Number of inactive pages
- `ubc_lru_page_count`—Number of UBC LRU pages

The following example shows the current value of the `vm_page_free_count` kernel variable:

```
(dbx) print vm_page_free_count
336
```

See Chapter 4 for information on managing memory resources.

2.5 Gathering General Disk Information

Use the following commands to gather general information about disks:

- The `iostat` command displays I/O statistics for disks, the CPU, and terminals. See Section 2.5.1.
- The `dbx nchstats` structure reports namei cache statistics. See Section 2.5.2.
- The `dbx vm_perfsum` structure reports UBC statistics, including the number of pages of memory that the UBC is using. See Section 2.5.3.
- The `dbx` debugger's `xpt_qhead`, `ccmn_bp_head`, and `xpt_cb_queue` structures report Common Access Method (CAM) statistics. See Section 2.5.4.

The following sections describe these commands in detail. You can also use the `diskx` exerciser to test disk drivers. See `diskx`(8).

2.5.1 Using iostat to Display Disk Usage

The `iostat` command reports I/O statistics for terminals, disks, and the CPU. The first line of the output is the average since boot time, and each subsequent report is for the last interval.

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

```
# iostat 1
      tty     rz1      rz2      rz3      cpu
 tin tout bps tps  bps tps  bps tps  us ni sy id
   0    3   3   1    0   0    8   1  11 10 38 40
   0   58   0   0    0   0    0   0  46  4 50  0
   0   58   0   0    0   0    0   0  68  0 32  0
   0   58   0   0    0   0    0   0  55  2 42  0
```

The `iostat` command reports I/O statistics that you can use to diagnose disk I/O performance problems. For example, the command displays the following information:

- For each disk, (`rz`*n*), the number of bytes (in thousands) 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 higher priority (`us`), in user mode running processes at a lowered priority (`ni`), in system mode (`sy`), and idle (`id`). This information enables you to determine how disk I/O is affecting the CPU.

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. The 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.

  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.
- 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.

See Chapter 5 for more information on how to improve disk performance.

2.5.2 Checking the namei Cache With dbx nchstats

The namei cache is used by UFS, AdvFS, CD-ROM File System (CDFS), and NFS to store recently used file system pathname/inode number pairs. It also stores inode 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 the namei cache, use the `dbx` debugger and look at the `nchstats` data structure. In particular, look at the `ncs_goodhits`, `ncs_neghits`, and `ncs_misses` 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`, and `ncs_misses`).

Consider the following example:

```
(dbx) print nchstats
struct {
    ncs_goodhits = 9748603   –found a pair
    ncs_neghits = 888729     –found a pair that didn’t exist
    ncs_badhits = 23470
    ncs_falsehits = 69371
    ncs_miss = 1055430       –did not find a pair
    ncs_long = 4067          –name was too long to fit in the cache
    ncs_pass2 = 127950
    ncs_2passes = 195763
    ncs_dirscan = 47
}
(dbx)
```

See Chapter 5 for information on how to improve the namei cache hit rate and lookup speeds.

2.5.3 Checking the UBC With dbx vm_perfsum

To check the Unified Buffer Cache (UBC), use the dbx debugger to examine the vm_perfsum data structure. In particular, look at the vpf_pgiowrites field (number of I/O operations for pageouts generated by the page stealing daemon) and the vpf_ubcalloc field (number of times the UBC had to allocate a page from the virtual memory free page list to satisfy memory demands).

Consider the following example:

```
(dbx) print vm_perfsum
struct {
    vpf_pagefaults = 493749
    vpf_kpagefaults = 3851
    vpf_cowfaults = 144197
    vpf_cowsteals = 99541
    vpf_zfod = 65590
    vpf_kzfod = 3846
    vpf_pgiowrites = 863
    vpf_pgwrites = 1572
    vpf_pgioreads = 187350
    vpf_pgreads = 17228
    vpf_swapreclaims = 0
    vpf_taskswapouts = 297
    vpf_taskswapins = 272
    vpf_vmpagesteal = 0
    vpf_vmpagewrites = 843
    vpf_vmpagecleanrecs = 1270
    vpf_vplmsteal = 18
    vpf_vplmstealwins = 16
    vpf_vpseqdrain = 0
    vpf_ubchit = 398
    vpf_ubcalloc = 21683
    vpf_ubcpushes = 0
    vpf_ubcpagepushes = 0
    vpf_ubcdirtywra = 0
    vpf_ubcreclaim = 0
    vpf_ubcpagesteal = 7071
    vpf_ubclookups = 364856
    vpf_ubclookuphits = 349473
    vpf_reactivate = 17352
    vpf_allocatedpages = 5800
    vpf_vmwiredpages = 437
    vpf_ubcwiredpages = 0
    vpf_mallocpages = 1115
    vpf_totalptepages = 207
    vpf_contigpages = 3
    vpf_rmwiredpages = 0
    vpf_ubcpages = 2090
    vpf_freepages = 918
    vpf_vmcleanpages = 213
    vpf_swapspace = 7996
}
(dbx)
```

The `vpf_ubcpages` field gives the number of pages of physical memory that the UBC is using to cache file data. If the UBC is using significantly more than half of physical memory and the paging rate is high (`vpf_pgiowrites` field), you may want to reduce the amount of memory available to the UBC to reduce paging. The default value of the `ubc-maxpercent` attribute is 100 percent of physical memory. Decrease this value only by increments of 10. However, reducing the value of the `ubc-maxpercent` attribute may degrade file system performance.

You can also monitor the UBC by examining the `ufs_getapage_stats` kernel data structure. To calculate the hit rate, divide the value for `read_hits` by the value for `read_looks`. A good hit rate is a rate above 95 percent.

Consider the following example:

```
(dbx) print ufs_getapage_stats
struct {
    read_looks = 2059022
    read_hits = 2022488
    read_miss = 36506
}
(dbx)
```

In the previous example, the hit rate is approximately 98 percent.

You can also check the UBC by examining the `vm_tune` data structure and the `vt_ubcseqpercent` and `vt_ubcseqstartpercent` fields. These values are used to prevent a large file from completely filling the UBC, which limits the amount of memory available to the virtual memory subsystem.

Consider the following example:

```
(dbx) print vm_tune
struct {
    vt_cowfaults = 4
    vt_mapentries = 200
    vt_maxvas = 1073741824
    vt_maxwire = 16777216
    vt_heappercent = 7
    vt_anonklshift = 17
    vt_anonklpages = 1
    vt_vpagemax = 16384
    vt_segmentation = 1
    vt_ubcpagesteal = 24
    vt_ubcdirtypercent = 10
    vt_ubcseqstartpercent = 50
    vt_ubcseqpercent = 10
    vt_csubmapsize = 1048576
    vt_ubcbuffers = 256
    vt_syncswapbuffers = 128
    vt_asyncswapbuffers = 4
    vt_clustermap = 1048576
    vt_clustersize = 65536
    vt_zone_size = 0
    vt_kentry_zone_size = 16777216
    vt_syswiredpercent = 80
    vt_inswappedmin = 1
}
```

When copying large files, the source and destination objects in the UBC will grow very large (up to all of the available physical memory). Reducing the value of the `vm-ubcseqpercent` attribute decreases the number of UBC pages that will be used to cache a large sequentially accessed file. The value represents the percentage of UBC memory that a sequentially accessed file can consume before it starts reusing UBC memory. The value imposes a resident set size limit on a file.

See Chapter 4 for information on how to tune the UBC.

2.5.4 Monitoring CAM Data Structures With dbx

The operating system uses the Common Access Method (CAM) as the operating system interface to the hardware. CAM maintains the xpt_qhead, ccmn_bp_head, and xpt_cb_queue data structures as follows:

- xpt_qhead—Contains information about the current size of the buffer pool free list (xpt_nfree), the current number of processes waiting for buffers (xpt_wait_cnt), and the total number of times that processes had to wait for free buffers (xpt_times_wait).
- ccmn_bp_head—Provides statistics on the buffer structure pool. This pool is used for raw I/O to disk. Some database applications with their own file system use the raw device instead of UFS. The information provided is the current size of the buffer structure pool (num_bp) and the wait count for buffers (bp_wait_cnt).
- xpt_cb_queue—Contains the actual link list of the I/O operations that have been completed and are waiting to be passed back to the peripheral drivers (cam_disk or cam_tape, for example).

The following examples use the dbx debugger to examine these three data structures:

```
(dbx) print xpt_qhead
struct {
    xws = struct {
        x_flink = 0xffffffff81f07400
        x_blink = 0xffffffff81f03000
        xpt_flags = 2147483656
        xpt_ccb = (nil)
        xpt_nfree = 300
        xpt_nbusy = 0
    }
    xpt_wait_cnt = 0
    xpt_times_wait = 2
    xpt_ccb_limit = 1048576
    xpt_ccbs_total = 300
    x_lk_qhead = struct {
        sl_data = 0
        sl_info = 0
        sl_cpuid = 0
        sl_lifms = 0
    }
}
(dbx) print ccmn_bp_head
struct {
    num_bp = 50
    bp_list = 0xffffffff81f1be00
    bp_wait_cnt = 0
}
(dbx) print xpt_cb_queue
struct {
    flink = 0xfffffc00004d6828
    blink = 0xfffffc00004d6828
    flags = 0
    initialized = 1
    count = 0
    cplt_lock = struct {
        sl_data = 0
        sl_info = 0
        sl_cpuid = 0
        sl_lifms = 0
    }
}
(dbx)
```

If the values for `xpt_wait_cnt` or `bp_wait_cnt` are nonzero, 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 CAM's I/O attributes (see Chapter 5).

The `count` parameter in `xpt_cb_queue` is the number of I/O operations that have been completed and are ready to be passed back to a peripheral device driver. Normally, the value of `count` should be 0 or 1. If the value is greater than 1, it may indicate either a problem or a temporary situation in which a large number of I/O operations are completing simultaneously. If repeated monitoring demonstrates that the value is consistently greater than 1, one or more subsystems may require tuning.

2.6 Gathering UFS Information

Use the following commands to gather information about the UNIX file system (UFS):

- The `dumpfs` command displays information about UFS file systems. See Section 2.6.1.
- The `dbx ufs_clusterstats` structure reports cluster read and write transfer statistics. See Section 2.6.2.
- The `dbx bio_stats` structure reports information about the metadata buffer cache. See Section 2.6.3.

The following sections describe these commands in detail.

In addition, you can use the `fsx` exerciser to test UFS and AdvFS file systems. See `fsx`(8) for information.

2.6.1 Using dumpfs to Display UFS Information

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:

```
# dumpfs /dev/rrz3g | more
magic   11954   format  dynamic time    Tue Sep 14 15:46:52 1996
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 held back from 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.

See Chapter 5 for more information about improving disk I/O performance.

2.6.2 Checking UFS Clustering With dbx ufs_clusterstats

To check UFS by using the `dbx` debugger, examine the `ufs_clusterstats` data structure to see how efficiently the system is performing cluster read and write transfers. You can examine the cluster reads and writes separately with the `ufs_clusterstats_read` and `ufs_clusterstats_write` data structures.

The following example shows a system that is not clustering efficiently:

```
(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.

See Chapter 5 for information on how to tune a UFS file system.

2.6.3 Checking the Metadata Buffer Cache With dbx bio_stats

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` debugger to examine the `bio_stats` data structure.

Consider the following example:

```
(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 Chapter 4 for information on how to tune the metadata buffer cache.

2.7 Gathering AdvFS Information

Use the following commands to gather information about the Advanced File System (AdvFS):

- The `advsstat` command displays AdvFS performance statistics. See Section 2.7.1.
- The `advscan` command identifies which disks are in a file domain. See Section 2.7.2.
- The `showfdmn` utility displays information about AdvFS file domains. See Section 2.7.3.
- The `showfile` command displays information about AdvFS filesets. See Section 2.7.4.
- The `showfsets` command displays information about the filesets in a file domain. See Section 2.7.5.

The following sections describe these commands in detail.

In addition, you can use the `fsx` exerciser to test AdvFS and UFS file systems. See `fsx`(8) for more information.

2.7.1 Using advfsstat to Display AdvFS Performance Information

The `advfsstat` command displays various AdvFS performance statistics. The command reports information in units of one disk block (512 bytes) for each interval of time (the default is one second).

Use the `advfsstat` command to monitor the performance of AdvFS domains and filesets. Use this command to obtain detailed information, especially if the `iostat` command output indicates a disk bottleneck.

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. You can use the `-i` option to output information at specific time intervals.

For example:

```
# advfsstat -v 2 test_domain
vol1
  rd   wr  rg  arg  wg  awg  blk  wlz  rlz  con  dev
  54   0  48  128   0    0    0    1   0     0   65
```

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.

Consolidating reads and writes improves performance. 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.

The I/O queue values (`blk` to `dev`) can indicate potential performance issues. 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 this 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.

If an application is I/O bound, you may be able to eliminate the problem by adding more disks to the domain or by striping disks. If the values for both the consolidate queue (`con`) and the device queue (`dev`) are large during periods of poor performance, you may want to increase the value of the `AdvfsMaxDevQLen` attribute. See Section 5.6.2.6 for information about modifying the attribute.

You can monitor the type of requests that applications are issuing by using the `advfsstat` command's `-f` flag 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.

The following example shows the startup, running, and completion times for an application:

```
# 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.

2.7.2 Using advscan to Identify Disks in an AdvFS File Domain

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 around in a way that has changed device numbers, or have lost track of where the domains are. You can also use this command for repair if you delete the `/etc/fdmns` directory, delete a directory domain under `/etc/fdmns`, or delete some links from a domain directory under `/etc/fdmns`.

The `advscan` command accepts a list of volumes or disk groups and searches all partitions and volumes in each. It determines which partitions on a disk are part of an AdvFS file domain. You can run 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:

```
# advscan rz0 rz5

Scanning disks  rz0 rz5
Found domains:

usr_domain
                Domain Id       2e09be37.0002eb40
                Created         Thu Jun 23 09:54:15 1996

                Domain volumes          2
                /etc/fdmns links        2

                Actual partitions found:
                                        rz0c
                                        rz5c
```

For the following example, the rz6 file domains were removed from /etc/fdmns. The advscan command scans device rz6 and re-creates the missing domains.

```
# advscan -r rz6

Scanning disks  rz6
Found domains:

*unknown*
                Domain Id       2f2421ba.0008c1c0
                Created         Mon Jan 23 13:38:02 1996

                Domain volumes          1
                /etc/fdmns links        0

                Actual partitions found:
                                        rz6a*
*unknown*
                Domain Id       2f535f8c.000b6860
                Created         Tue Feb 28 09:38:20 1996

                Domain volumes          1
                /etc/fdmns links        0

                Actual partitions found:
                                        rz6b*

Creating /etc/fdmns/domain_rz6a/
        linking rz6a

Creating /etc/fdmns/domain_rz6b/
        linking rz6b
```

See advscan(8) for more information.

2.7.3 Using showfdmn to Display AdvFS File Domain Information

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 usr file domain:

```
% showfdmn usr

               Id              Date Created  LogPgs  Domain Name
2b5361ba.000791be  Tue Jan 12 16:26:34 1996     256  usr
```

```
Vol   512-Blks      Free  % Used  Cmode  Rblks  Wblks  Vol Name
 1L     820164    351580     57%     on    256    256  /dev/rz0d
```

See showfdmn(8) for more information about the output of the command.

2.7.4 Using showfile to Display AdvFS File Information

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-specific attributes for all of the files in the current working directory:

```
# showfile *

       Id  Vol  PgSz  Pages  XtntType  Segs  SegSz    I/O  Perf File
  22a.001    1    16      1    simple    **     **  async  50%  Mail
    7.001    1    16      1    simple    **     **  async  20%  bin
  1d8.001    1    16      1    simple    **     **  async  33%  c
 1bff.001    1    16      1    simple    **     **  async  82%  dxMail
  218.001    1    16      1    simple    **     **  async  26%  emacs
  1ed.001    1    16      0    simple    **     **  async 100%  foo
  1ee.001    1    16      1    simple    **     **  async  77%  lib
  1c8.001    1    16      1    simple    **     **  async  94%  obj
  23f.003    1    16      1    simple    **     **  async 100%  sb
 170a.008    1    16      2    simple    **     **  async  35%  t
    6.001    1    16     12    simple    **     **  async  16%  tmp
```

The I/O column specifies whether the I/O operation is synchronous or asynchronous.

The following example of the showfile command shows the attributes and extent information for the tutorial file, which is a simple file:

```
# 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.

2.7.5 Using showfsets to Display AdvFS Filesets in a File Domain

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:

```
# showfsets dmn

mnt
          Id           : 2c73e2f9.000f143a.1.8001
          Clone is     : mnt_clone
          Files        :       79,  limit =     1000
          Blocks  (1k) :      331,  limit =    25000
          Quota Status : user=on  group=on

mnt_clone
          Id           : 2c73e2f9.000f143a.2.8001
          Clone of     : mnt
          Revision     : 1
```

See `showfsets`(8) for information about the options and output of the command.

2.8 Gathering LSM Information

Use the following commands to gather information about a Logical Storage Manager (LSM) configuration:

- The `volprint` command displays comprehensive information about an LSM configuration. See Section 2.8.1.
- The `volstat` utility reports I/O performance statistics for an LSM configuration. See Section 2.8.2.
- The `voltrace` utility tracks I/O operations on LSM volumes. See Section 2.8.3.
- The `volwatch` script monitors an LSM configuration for failures. See Section 2.8.4.
- The `dxlsm` graphical user interface (GUI) reports comprehensive information about an LSM configuration. See Section 2.8.5.

The following sections describe these commands in detail.

2.8.1 Using volprint to Display LSM Configuration Information

The `volprint` command displays information from records in the LSM configuration database. You can select the 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.

Use the `volprint` command to display disk group, disk media, volume, plex, and subdisk records. Use the `voldisk list` to display disk access records, or physical disk information.

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

```
# volprint -ht voldev1
DG NAME        GROUP-ID
DM NAME        DEVICE       TYPE     PRIVLEN  PUBLEN   PUBPATH
V  NAME        USETYPE      KSTATE   STATE    LENGTH   READPOL  PREFPLEX
PL NAME        VOLUME       KSTATE   STATE    LENGTH   LAYOUT   ST-WIDTH MODE
SD NAME        PLEX         PLOFFS   DISKOFFS LENGTH   DISK-NAME    DEVICE

v  voldev1     fsgen        ENABLED  ACTIVE   804512   SELECT   -
pl voldev1-01  voldev1      ENABLED  ACTIVE   804512   CONCAT   -        RW
sd rz8-01      voldev1-01   0        0        804512   rz8          rz8
pl voldev1-02  voldev1      ENABLED  ACTIVE   804512   CONCAT   -        RW
sd dev1-01     voldev1-02   0        2295277  402256   dev1         rz9
sd rz15-02     voldev1-02   402256   2295277  402256   rz15         rz15
```

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

2.8.2 Using volstat to Display LSM Performance Information

The `volstat` command 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.

The amount of information displayed depends on which options you specify to `volstat`. 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, `volstat` displays statistics for the default disk group (`rootdg`).

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

The following example uses the `volstat` command to display statistics on LSM volumes:

```
# volstat
OPERATIONS       BLOCKS        AVG TIME(ms)
TYP NAME        READ   WRITE    READ    WRITE   READ   WRITE
vol archive      865     807    5722     3809   32.5    24.0
vol home        2980    5287    6504    10550   37.7   221.1
vol local      49477   49230  507892   204975   28.5    33.5
vol src        79174   23603  425472   139302   22.4    30.9
vol swapvol    22751   32364  182001   258905   25.3   323.2
```

See `volstat`(8) for more information about command output.

2.8.3 Using voltrace to Display LSM I/O Operation Information

The `voltrace` command reads an event log (`/dev/volevent`) and prints formatted event log records to standard output. Using `voltrace`, you can set event trace masks to determine which type of events will be tracked. For example, you can trace I/O events, configuration changes, or I/O errors.

The following example uses the `voltrace` command to display status on all new events:

```
# voltrace -n -e all
18446744072623507277 IOTRACE 439: req 3987131 v:rootvol p:rootvol-01 \
  d:root_domain s:rz3-02 iot write lb 0 b 63120 len 8192 tm 12
18446744072623507277 IOTRACE 440: req 3987131 \
  v:rootvol iot write lb 0 b 63136 len 8192 tm 12
```

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

2.8.4 Using volwatch to Monitor LSM Failures

The `volwatch` shell script is automatically started when you install LSM. This script sends mail to root if certain LSM configuration events occur, such as a plex detach caused by a disk failure. The script sends mail to root by default. You also can specify another mail recipient.

See `volwatch`(8) for more information.

2.8.5 Using dxlsm to Display LSM Configuration Information

The LSM graphical user interface (GUI), `dxlsm`, includes an Analyze menu that allows you to display statistics about volumes, LSM disks, and subdisks. The information is displayed graphically, using colors and patterns on the disk icons, and numerically, using the `Analysis Statistics` form. You can use the `Analysis Parameters` form to customize the displayed information.

See the *Logical Storage Manager* manual and `dxlsm`(8X) for more information about `dxlsm`.

2.9 Gathering Network Information

Use the following commands to gather network performance information:

- The `netstat` command displays network statistics. See Section 2.9.1.
- The `nfsstat` command displays network and NFS statistics. See Section 2.9.2.
- The `sobacklog_hiwat` attribute reports pending requests to a server socket. See Section 2.9.3.
- The `sobacklog_drops` attribute reports the number of backlog drops that exceed the limit. See Section 2.9.3.
- The `somaxconn_drops` attribute reports the number of drops that exceed the `somaxconn` limit. See Section 2.9.3.
- The `ps` command displays information about idle threads. See Section 6.2.4.

The following sections describe these commands in detail.

In addition, you can use the following commands to obtain network information:

- The `tcpdump` command monitors network interface packets. See `tcpdump`(8).
- The `traceroute` command displays a packet's route to a network host. See `traceroute`(8).
- The `ping` command determines if a host can be reached on a network. See `ping`(8).
- The `nfswatch` command monitors an NFS server. See `nfswatch`(8).

2.9.1 Using netstat to Display Network Information

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

- 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.
- If the `netstat -m` command shows several requests for memory delayed or denied, this means that your system was temporarily short of physical memory.
- 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.

- 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:

  ```
  # netstat -an | grep TCP | awk '{print $6}' | sort | uniq -c
     1 CLOSED
    18 CLOSE_WAIT
   380 ESTABLISHED
    74 LISTEN
     9 TIME_WAIT
  ```

- 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 look for bad checksums and full sockets.
- Use the `netstat -rs` to obtain routing statistics.

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

The following example shows the output produced by the `netstat -i` command:

```
# 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:

```
# netstat -is

ln0 Ethernet counters at Fri Jan 14 16:57:36 1996

        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 `netstat -s` command displays the following statistics for each protocol:

```
# 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 information about the output produced by the various options supported by the `netstat` command.

2.9.2 Using nfsstat to Display Network and NFS Information

The `nfsstat` command displays statistical information about the Network File System (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:

```
# 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 6 for information on how to tune your system to avoid timeouts.

If you are attempting to monitor an experimental situation with `nfsstat`, reset the NFS counters to zero 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.

2.9.3 Checking Socket Listen Queue Statistics With sysconfig

You can determine whether you need to increase the socket listen queue limit by using the `sysconfig -q socket` command to display the values of the following attributes:

- `sobacklog_hiwat`

  Allows you to monitor the maximum number of pending requests to any server socket in the system. The initial value is 0.

- `sobacklog_drops`

  Allows you to monitor the number of times the system dropped a received SYN packet because the number of queued SYN_RCVD connections for a socket equaled the socket's backlog limit. The initial value is 0.

- `somaxconn_drops`

  Allows you to monitor the number of times the system dropped a received SYN packet because the number of queued SYN_RCVD connections for the socket equaled the upper limit on the backlog length (`somaxconn` attribute). The initial value is 0.

  DIGITAL recommends 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 Chapter 6 for information on tuning socket listen queue limits.

2.9.4 Using ps to Display Idle Thread Information

On a client system, the `nfsiod` daemons spawn 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:

```
#
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. If your output shows that few threads are sleeping, you may be able to improve NFS performance by increasing the number of threads. See Chapter 6, `nfsiod`(8), and `nfsd`(8) for more information.

2.10 Gathering Profiling and Debugging Information

For information about the application and kernel profiling and debugging tools that are described in Table 2–7, see the *Programmer's Guide*, the *Kernel Debugging* manual, and the reference pages associated with the tools.

2.11 Modifying the Kernel

Kernel variables, including system attributes and parameters, determine the behavior of the DIGITAL UNIX operating system and subsystems. When you install the operating system or add optional subsystems, the kernel variables are set to their default values. Modifying the values of certain kernel variables may improve system performance. Some kernel variables are used only to monitor the current state of the system.

You can display and modify kernel variable values by using various methods. You can modify some variables by using all methods, but in some cases, you must use a particular method to modify a variable.

Because you can use various methods to assign values to kernel variables, the system uses the following hierarchy to determine which value to use:

- Run-time values

  Run-time kernel variable values are effective immediately. Not all variables can be tuned at run time. Use the Kernel Tuner, `dxkerneltuner`, or the `sysconfig -r` command to make run-time modifications to attributes that support this feature. When the system reboots, these run-time modifications are lost, and the attributes revert to their permanently assigned values. See Section 2.11.2 and Section 2.11.3 for more information.

You also can use the `dbx assign` command or the `dbx patch` command to make modifications to variables in the running kernel. If you use the `dbx assign` command, the modifications are lost when you reboot the system. If you use the `dbx patch` command, the modifications are lost when you rebuild the kernel. See Section 2.11.1 for more information.

- Permanent attribute values

  The `sysconfigtab` subsystem configuration database file describes the various subsystems and their attribute values. Use the Kernel Tuner or the `sysconfigdb` command to assign values to attributes in the `sysconfigtab` file. Do not manually edit the `sysconfigtab` file.

  See Section 2.11.2 and Section 2.11.4 for more information.

- Permanent parameter values

  The `/usr/sys/conf/`*`SYSTEM`* configuration file describes system parameters. You can edit the file to modify the values assigned to the parameters. You must rebuild the kernel and reboot the system to use the new parameter values.

  Because some system attributes have corresponding system parameters, the values permanently assigned to attributes supersede the values permanently assigned to their corresponding parameters. If possible, modify an attribute instead of its corresponding parameter.

  See Section 2.11.5 for more information.

The following sections describe how to display and modify kernel variables, attributes, and parameters. See the *System Administration* manual for detailed information about kernel variables, attributes, and parameters.

2.11.1 Using dbx to Display and Modify Run-Time Kernel Variables

Use the `dbx` command to examine source files, control program execution, display the state of the program, and debug at the machine-code level. To examine the values of kernel variables and data structures, use the `dbx print` command and specify the data structure or variable to examine.

An example of the `dbx print` command is as follows:

```
# dbx –k /vmunix /dev/mem
```

```
(dbx)  print vm_page_free_count
248
(dbx)
```

```
# dbx –k /vmunix /dev/mem

(dbx)  print somaxconn
1024
(dbx)
```

```
# dbx –k /vmunix /dev/mem

(dbx)  print vm_perfsum
struct {
    vpf_pagefaults = 1689166
    vpf_kpagefaults = 13690
    vpf_cowfaults = 478504
    vpf_cowsteals = 638970
    vpf_zfod = 255372
    vpf_kzfod = 13654
    vpf_pgiowrites = 3902

⋮

    vpf_vmwiredpages = 440
    vpf_ubcwiredpages = 0
    vpf_mallocpages = 897
    vpf_totalptepages = 226
    vpf_contigpages = 3
    vpf_rmwiredpages = 0
    vpf_ubcpages = 2995
    vpf_freepages = 265
    vpf_vmcleanpages = 237
    vpf_swapspace = 7806
}
(dbx)
```

Use the `dbx patch` command to modify the run-time values of some kernel variables. Note that the values you assign by using the `dbx patch` command are temporary and are lost when you rebuild the kernel.

An example of the `dbx patch` command is as follows:

```
# dbx –k /vmunix /dev/mem

(dbx)  patch somaxconn = 32767
32767
(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.

2.11.2 Using the Kernel Tuner to Display and Modify Attributes

Use the Kernel Tuner (`dxkerneltuner`), provided by the Common Desktop Environment's (CDE) graphical user interface, to display the current and permanent values for attributes, modify the run-time values (if supported), and modify the permanent values.

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, allowing you to select a subsystem and generate a display of the subsystem's attributes and values.

2.11.3 Using the sysconfig Command to Display and Modify Run-Time Attributes

Use the `sysconfig` command to display the configured subsystems, attribute values, and other attribute information. The command also allows you to modify the run-time values of attributes that support this feature.

Use the `sysconfig -s` command to list the subsystems that are configured in your system. An example of the `sysconfig -s` command is as follows:

```
# sysconfig -s
Cm: loaded and configured
Generic: loaded and configured
Proc: loaded and configured
⋮
Xpr: loaded and configured
Rt: loaded and configured
Net: loaded and configured
#
```

Use the `sysconfig -q` command and specify a subsystem to display the run-time values of the subsystem attributes. An example of the `sysconfig -q` command is as follows:

```
# sysconfig -q vfs
vfs:
name-cache-size = 32768
name-cache-hash-size = 1024
buffer-hash-size = 512
:
max-ufs-mounts = 1000
vnode-deallocation-enable = 1
pipe-maxbuf-size = 65536
pipe-single-write-max = -1
pipe-databuf-size = 8192
pipe-max-bytes-all-pipes = 81920000
noadd-exec-access = 0
#
```

If an attribute is not defined in the `sysconfigtab` database file, the `sysconfig -q` command returns the default value of attribute.

To display the minimum and maximum values for an attribute, use the `sysconfig -Q` command and specify the subsystem. An example of the `sysconfig -Q` command is as follows:

```
# sysconfig -Q ufs
ufs:
inode-hash-size - type=INT op=CQ min_val=0 max_val=2147483647
create-fastlinks - type=INT op=CQ min_val=0 max_val=2147483647
ufs-blkpref-lookbehind -
 type=INT op=CQ min_val=0 max_val=2147483647
nmount - type=INT op=CQ min_val=0 max_val=2147483647
#
```

To modify the run-time value of an attribute, use the `sysconfig -r` command and specify the subsystem, the attribute, and the attribute value. Only some attributes support run-time modifications. An example of the `sysconfig -r` command is as follows:

```
# sysconfig -r socket somaxconn=1024
somaxconn: reconfigured
#
```

See the *System Administration* manual and `sysconfig`(8) for more information.

2.11.4 Using the sysconfigdb Command to Modify Attributes

Use the `sysconfigdb` command to assign new values to attributes in the `sysconfigtab` database file. Do not manually edit the `sysconfigtab` database file.

After you use the `sysconfigdb` command, reboot the system or invoke the `sysconfig -r` command to use the new attribute values.

See the *System Administration* manual and `sysconfigdb`(8) for more information.

2.11.5 Modifying Parameters in the System Configuration File

Use the `/usr/sys/conf/`*`SYSTEM`* configuration file to specify values for kernel parameters. You can edit the file to modify the values currently assigned to the parameters or to add parameters. You must rebuild the kernel and reboot the system to use the new parameter values.

Some kernel attributes have corresponding kernel parameters, but the values permanently assigned to attributes supersede the values permanently assigned to their corresponding parameters in the system configuration file. If possible, always modify an attribute instead of its corresponding parameter.

See the *System Administration* manual for descriptions of some parameters and information about modifying the system configuration file and rebuilding the kernel. See Appendix B for a list of attributes that have corresponding parameters.

3 Optimizing Applications and CPU Performance

This chapter describes how to optimize CPU resources and applications for high performance.

3.1 Configuring CPU Resources

You must configure enough CPU power in your system to meet the performance needs of your users and applications. In addition, you may be able to improve performance by optimizing the CPU and your applications.

A system must be able to efficiently allocate the available CPU cycles among competing processes. In addition to single-CPU systems, DIGITAL supports multiprocessing systems and processors with different speeds.

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, World Wide Web (WWW) 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. However, increasing the number of processors may increase the demands on your I/O and memory subsystems and could cause bottlenecks. If your system is metadata-intensive (that is, it opens large numbers of small files and accesses them repeatedly), you may gain an additional performance benefit if you add Prestoserve or use a write-back cache when you add more processors. See Chapter 5 for information about Prestoserve and write-back caches.

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.

The `iostat` and `vmstat` commands let you monitor the memory, CPU, and I/O consumption on your system. The `cpustat` extension to the `kdbx` debugger allows application developers to monitor the time spent in user mode, system mode, and kernel mode on each of the processors. This information can help application developers determine how effectively they are achieving parallelism across the system. See Chapter 2 for information about using tools to monitor performance.

3.2 Identifying CPU Bottlenecks

Use the `vmstat` command to determine CPU usage as follows:

- 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
- A low percentage of idle time is the primary indication of a CPU bottleneck.
- 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 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 processing power, or excessive memory latency or cache missing.

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

Use the `kdbx cpustat` extension to display statistics about CPU use, especially for multiprocessing systems. Statistics include 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)

See Chapter 2 for information about monitoring systems.

3.3 Optimizing CPU Resources

After you configure the appropriate number of CPUs in your system, you may be able to improve system performance by optimizing your CPU resources. Before optimizing the CPU, ensure that the virtual memory or I/O subsystems are not the cause of poor performance. If optimizing the CPU does not solve the performance problem, you must upgrade your CPU to a faster processor or use multiprocessing.

To optimize your CPU resources, use the following methods:

- Use the Class Scheduler to allocate CPU resources

  The Class Scheduler allows you 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.

  To use class scheduling, group together processes into classes and assign each class a percentage of CPU time. You can display statistics on the actual CPU usage for a class. You can also manually assign a class to any process.

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

- 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.

- Schedule jobs at different times (use the `at` and `cron` commands) or when the load level permits (use the `batch` command)

- Increase the program size limits

  Extremely large programs may run more efficiently if you increase the values of the following system configuration file parameters that control program size limits:

  - `dfldsiz`—Default data segment size limit
  - `maxdsiz`—Maximum data segment size limit
  - `dflssiz`—Default stack size limit
  - `maxssiz`—Maximum stack size limit

  Some extremely large programs may not run unless these parameters are adjusted. For example, an inadequate `maxdsiz` size limit may produce the following error:

  ```
  Out of process memory...
  ```

The `limit` and `unlimit` commands can affect program size limits. See the *System Administration* manual for information on changing these parameter values.

- Reduce the size of the kernel

  You can reduce the static size of the kernel by deconfiguring any unnecessary subsystems. To do this, use the `setld -d` command. You can also minimize the number of kernel options for your system. See the *Installation Guide* for details.

- Ensure that lockmode is set to the appropriate value

  Specify 0 for UP (uniprocessing), 2 for symmetrical multiprocessing (SMP), and 1 or 3 for realtime. This can prevent system bottlenecks in the CPU.

- Optimize your applications

  You can use various compiler and linker optimization levels to generate more efficient user code. See the *Programmer's Guide* for more information on application optimization.

3.4 Identifying Application Bottlenecks

If an application is degrading system performance, use profiling to identify sections of code that consume large portions of execution time. In a typical program, most execution time is spent in relatively few sections of code. To improve performance, concentrate on improving the coding efficiency of those time-intensive sections. See the *Programmer's Guide* for more information on profiling.

3.5 Improving Application Performance

Well-written applications use CPU, memory, and I/O resources efficiently. You may be able to improve system and application performance by following these recommendations:

- Use the latest version of the operating system, compiler, firmware, and patches

  Check the software on your system to ensure that you are using the latest versions of the compiler and the operating system to build your application program. In general, new versions of a compiler perform advanced optimizations, and new versions of the operating system operate efficiently.

- Use 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.

- Ensure that the application runs without error

  Test your application program to ensure that it runs without errors. Whether you are porting an application from a 32-bit system to DIGITAL 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` flag or compile your program using the C compiler's `-check` flag to identify possible portability problems that you may need to resolve.

- Optimize 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.

- Prioritize applications

  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.

- Use 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.

3.6 Interprocess Communications Facilities

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. IPC is a functional interrelationship of several operating system subsystems. Elements are found in scheduling and networking.

In single-process programming, modules within a single process communicate with each other using global variables and function calls, with data passing between the functions and the callers. If you are programming by using separate processes with images in separate address spaces, use additional communication mechanisms.

The DIGITAL 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 about signals.
- Sockets—See the *Network Programmer's Guide* for information about sockets.
- Streams—See the *Programmer's Guide: STREAMS* for information about streams.
- X/Open Transport Interface (XTI)—See the *Network Programmer's Guide* for information about XTI.

You may be able to improve IPC performance by modifying the following attributes:

- `msg-mnb` (maximum number of bytes on queue)

  A process will be unable to send a message to a queue if the message will make the total number of bytes in that queue greater than the limit specified by `msg-mnb`. When the limit is reached, the process sleeps and waits for this condition to be resolved.

- `msg-tql` (number of system message headers)

  A process will be unable to send a message if the message will make the total number of message headers currently in the system greater than the limit specified by `msg-tql`. If the limit is reached, the process sleeps and waits for a message header to be freed.

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 increase the values of the `msg-mnb` and `msg-tql` attributes to diminish waiting.

You may also want to consider modifying several of the following IPC attributes:

- Message attributes:
  - `msg-max` (maximum message size)
  - `msg-mni` (number of message queue identifiers)
- Semaphore attributes:
  - `sem-mni` (number of semaphore identifiers)
  - `sem-msl` (number of semaphores per ID)
  - `sem-opm` (maximum number of operations per `semop` call)

  - `sem-ume` (maximum number of undo entries per process)
  - `sem-vmx` (semaphore maximum value)
  - `sem-aem` (adjust on exit maximum value)
- Shared memory attributes:
  - `shm-max` (maximum shared memory segment size)
  - `shm-min` (minimum shared memory segment size)
  - `shm-mni` (number of shared memory identifiers)
  - `shm-seg` (maximum attached shared memory segments per process)

  As a design consideration, consider whether you will get better performance by using threads instead of shared memory.

4
Configuring and Tuning Memory

This chapter describes how the DIGITAL UNIX operating system uses the physical memory installed in the system. This chapter also describes how to configure and tune virtual memory, swap space, and buffer caches. Many of the tuning tasks described in this chapter require you to modify system attributes. See Section 2.11 for more information.

4.1 Understanding Memory Management

The total amount of **physical memory** is determined by the capacity of the memory boards installed in your system. The system distributes this memory in 8-KB units called **pages**.

The system distributes pages of physical memory among three areas:

- **Wired memory**

  At boot time, the operating system and the Privileged Architecture Library (PAL) code wire a contiguous portion of physical memory in order to perform basic system operations. **Static wired memory** is reserved for operating system data and text, system tables, the metadata buffer cache, which temporarily holds recently accessed UNIX File System (UFS) and CD-ROM File System (CDFS) metadata, and the Advanced File System (AdvFS) buffer cache. Static wired memory cannot be reclaimed through paging. You can reduce the amount of static wired memory only by removing subsystems.

  In addition, the kernel uses **dynamically wired memory** for dynamically allocated data structures. User processes also wire memory for address space. The amount of dynamically wired memory varies according to the demand. The maximum amount is specified by the value of the `vm-syswiredpercent` attribute (the default is 80 percent of physical memory). Memory that is dynamically wired cannot be reclaimed through paging. You can reduce the amount of dynamically wired memory by allocating more kernel resources to processes (for example, by increasing the value of the `maxusers` attribute).

- **Virtual memory**

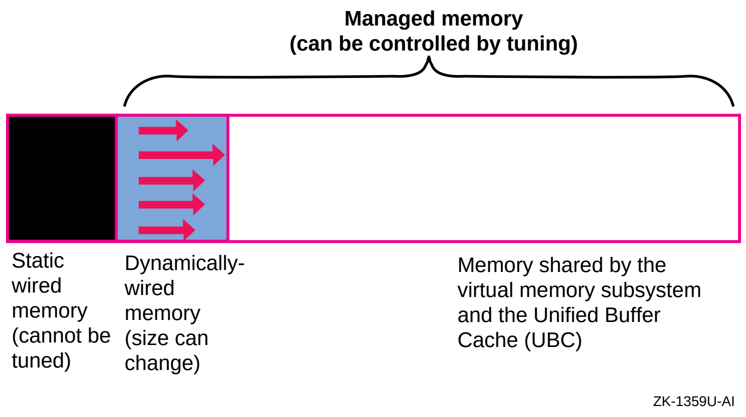
  The virtual memory subsystem uses a portion of physical memory to cache processes' most-recently accessed anonymous memory and file-backed memory. The subsystem efficiently allocates memory to competing processes and tracks the distribution of all the physical pages. This memory can be reclaimed through paging.

- **Unified Buffer Cache**

  The Unified Buffer Cache (UBC) uses a portion of physical memory to cache most-recently accessed file system data. The UBC contains actual file data for reads and writes and for page faults from mapped file regions and also AdvFS metadata. By functioning as a layer between the operating system and the storage subsystem, the UBC can decrease the number of disk operations. This memory can be reclaimed through paging.

Figure 4–1 shows how physical memory is used.

Figure 4–1: Physical Memory Usage


Managed memory
(can be controlled by tuning)
Static wired memory (cannot be tuned)
Dynamically-wired memory (size can change)
Memory shared by the virtual memory subsystem and the Unified Buffer Cache (UBC)
ZK-1359U-AI

The virtual memory subsystem and the UBC compete for the physical pages that are not wired. Pages are allocated to processes and to the UBC, as needed. When the demand for memory increases, the oldest (least-recently used) pages are reclaimed from the virtual memory subsystem and the UBC and reused. Various attributes control the amount of memory available to the virtual memory subsystem and the UBC and the rate of page reclamation. Wired pages are not reclaimed.

System performance depends on the total amount of physical memory and also the distribution of memory resources. DIGITAL UNIX allows you to control the allocation of memory (other than static wired memory) by modifying the values of system attributes. Tuning memory usually involves the following tasks:

- Increasing system resource allocation to improve application performance

- Modifying how the system allocates memory and the rate of page reclamation
- Modifying how file system data is cached in memory

You can also configure your swap space for optimal performance. However, to determine how to obtain the best performance, you must understand your workload characteristics, as described in Chapter 1.

4.2 Understanding Memory Hardware

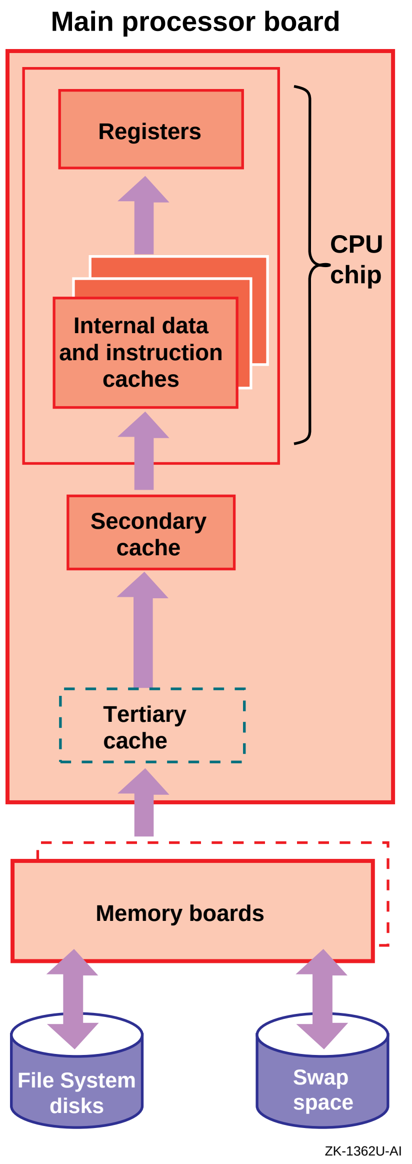
When programs are executed, the system moves data and instructions among various caches, physical memory, and disk swap space. Accessing the data and instructions occurs at different speeds, depending on the location. Table 4–1 describes the various hardware resources (in the order of fastest to slowest access time).

Table 4–1: Memory Management Hardware Resources

Resource	Description
CPU caches	Various caches reside in the CPU chip and 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.
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).

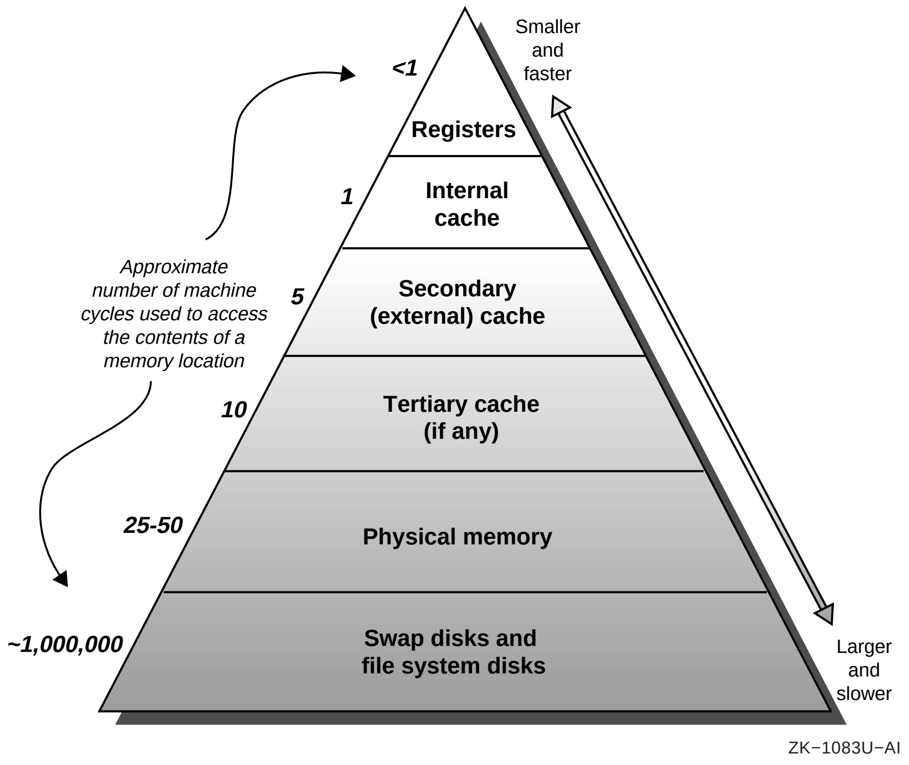
The hardware logic and the PAL code control much of the movement of addresses and data among the CPU cache, the secondary and tertiary caches, and physical memory. This movement is transparent to the operating system. Figure 4–2 shows an overview of how instructions and data are moved among various hardware components during program execution.

Figure 4–2: Moving Instructions and Data Through the Memory Hardware



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. Therefore, avoid paging and swapping operations, and applications should utilize caches when possible. Figure 4–3 shows the amount of time that it takes to access data and instructions from various hardware locations.

Figure 4–3: Time Consumed to Access Storage Locations



Smaller and faster
<1
Registers
1
Internal cache
5
Secondary (external) cache
Approximate number of machine cycles used to access the contents of a memory location
10
Tertiary cache (if any)
25-50
Physical memory
~1,000,000
Swap disks and file system disks
Larger and slower
ZK−1083U−AI

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

4.3 Understanding Virtual Memory

The virtual memory subsystem performs the following functions:

- Allocates memory to processes
- Tracks and manages all the pages in the system
- Uses paging and swapping to ensure that there is enough memory for processes to run and to cache file system I/O

The following sections describe these functions in detail.

4.3.1 Allocating Virtual Address Space to Processes

For each process, the `fork` system call performs the following tasks:

- 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.
- Allocates a contiguous block of **virtual address space**, which is the array of pages that an application can map into physical memory. Virtual address space is used for **anonymous memory** (memory used for the stack, heap, or `malloc` function) and for **file-backed memory** (memory used for program text or shared libraries). Pages of anonymous memory are paged in when needed and paged out when pages must be reclaimed. Pages of file-backed memory are paged in when needed and released when pages must be reclaimed.
- Creates one or more threads of execution. The default is one thread for each process. Multiprocessing systems support multiple process threads.

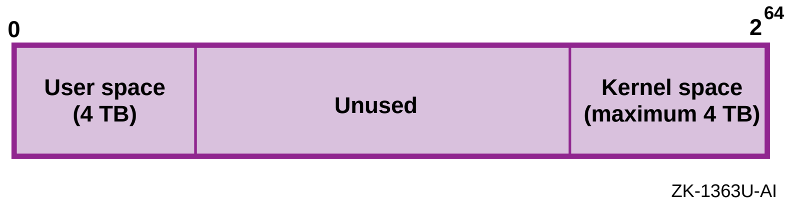
Because memory is limited, a process' entire virtual address space cannot be in physical memory at one time. However, a process can execute when only a portion of its virtual address space (its working set) is mapped to physical memory.

For each process, the virtual memory subsystem allocates a large amount of virtual address space but 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.

In addition, user space is sparsely populated with valid pages. Only valid pages are able to map to physical pages. The `vm-maxvas` attribute specifies the maximum amount of valid virtual address space for a process (that is, the sum of all the valid pages). The default is 128000 pages (1 GB).

Figure 4–4 shows the use of process virtual address space.

Figure 4–4: Virtual Address Space Usage

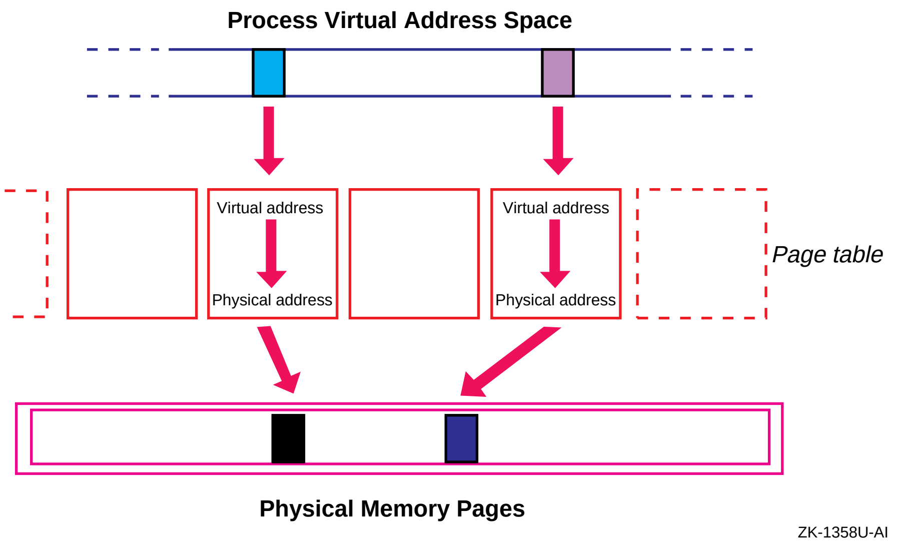


4.3.2 Translating Virtual Addresses to Physical Addresses

When a virtual page is touched or 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 4–5 shows the translation of a virtual address into a physical address.

Figure 4–5: 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 physical addresses. The working set is a subset of the resident set and represents a snapshot of the process' resident set.

4.3.3 Page Faulting

When a nonfile-backed virtual address is requested, the virtual memory subsystem locates the physical page and makes it available to the process. This process occurs at different speeds, depending on the location of the page (see Figure 4–3).

If a requested address is currently being used (active), 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.

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.

If a requested virtual address is being accessed for the first time, 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.

This is called a **zero-filled-on-demand page fault**.

If a requested virtual address has already been accessed, it will be in one of the following locations:

- The virtual memory subsystem's internal data structures

  If the physical address is located in the internal data structures (for example, the hash queue list or the page queue list), the virtual memory subsystem enters the virtual-to-physical address translation in the page table. This is called a **short page fault**.

- Swap space

  If the virtual address has already been accessed, but the physical page has been reclaimed, the page contents will be found in swap space. 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. This is called a **page-in page fault**.

If a process needs to modify a read-only virtual page, 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. This is called a **copy-on-write page fault**.

To improve process execution time and decrease the number of page faults, 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.

The virtual memory subsystem also uses **page coloring** to reduce execution time. If possible, the subsystem attempts to map a process' entire resident set into the secondary cache. If the entire task, text, and data are executed within the cache, addresses do not have to be fetched from physical memory.

The `private-cache-percent` attribute specifies the percentage of the cache that is reserved for anonymous (nonshared) memory. 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. This attribute is primarily used for benchmarking.

4.3.4 Managing and Tracking Pages

The virtual memory subsystem allocates physical pages to processes and the UBC, as needed. Because physical memory is limited, these pages must be periodically reclaimed so that they can be reused.

The virtual memory subsystem uses page lists to track the location and age of all the physical memory pages. 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 (the size of this list controls when page reclamation occurs)
- **Active list**—Pages that are being used by the virtual memory subsystem or the UBC

  To determine which pages should be reclaimed first, the page-stealer daemon identifies the oldest pages on the active list and designates these least-recently used (LRU) pages as follows:

  – Inactive pages are the oldest pages that are being used by the virtual memory subsystem.

  – UBC LRU pages are the oldest pages that are being used by the UBC.

Use the `vmstat` command or `dbx` 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.

As physical pages are allocated to processes and the UBC, the free list becomes depleted, and pages must be reclaimed in order to replenish the list. To reclaim pages, the virtual memory subsystem does the following:

- Prewrites the oldest dirty (modified) pages to swap space

- Uses paging to reclaim individual pages
- Uses swapping to suspend processes and reclaim a large number of pages

See Section 4.3.5, Section 4.3.6, Section 4.3.8, and Section 4.3.9 for more information about prewriting pages, paging, and swapping.

4.3.5 Prewriting Modified Pages

The virtual memory subsystem attempts to prevent a memory shortage by prewriting modified pages to swap space.

When the virtual memory subsystem anticipates that the pages on the free list will soon be depleted, it prewrites to swap space the oldest modified (dirty) inactive pages. The value of the `vm-page-prewrite-target` attribute determines the number of pages that the subsystem will prewrite and keep clean. The default value is 256 pages.

In addition, when the number of modified UBC LRU pages exceeds the value of the `vm-ubcdirtypercent` attribute, the virtual memory subsystem prewrites to swap space the oldest modified UBC LRU pages. The default value of the `vm-ubcdirtypercent` attribute is 10 percent of the total UBC LRU pages.

To minimize the impact of `sync` (steady state flushes) when prewriting UBC pages, the `ubc-maxdirtywrites` attribute specifies the maximum number of disk writes that the kernel can perform each second. The default value is 5.

See Section 4.7.13 for more information about prewriting dirty pages.

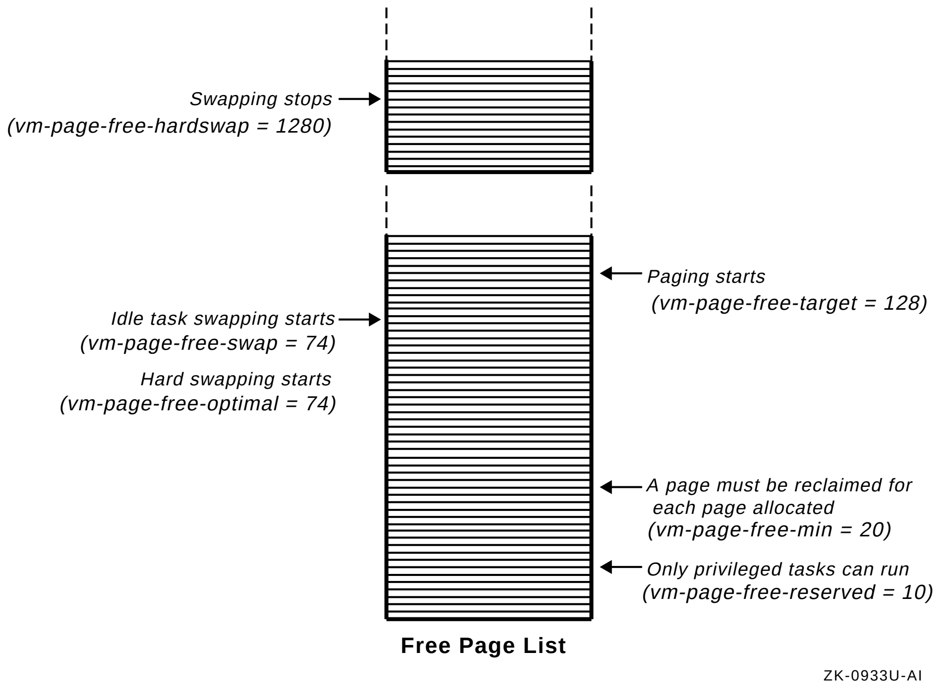
4.3.6 Using Attributes to Control Paging and Swapping

When the demand for memory depletes the free list, **paging** begins. The virtual memory subsystem takes the oldest inactive and UBC LRU pages, moves the contents of the modified pages to swap space, and puts the clean pages on the free list, where they can be reused.

If the free page list cannot be replenished by reclaiming individual pages, **swapping** begins. Swapping temporarily suspends processes and moves entire resident sets to swap space, which frees large amounts of physical memory.

The point at which paging and swapping start and stop depends on the values of some virtual memory subsystem attributes. Figure 4–6 shows the default values of these attributes.

Figure 4–6: Paging and Swapping Attributes – Default Values


Swapping stops
(vm-page-free-hardswap = 1280)
Paging starts
(vm-page-free-target = 128)
Idle task swapping starts
(vm-page-free-swap = 74)
Hard swapping starts
(vm-page-free-optimal = 74)
A page must be reclaimed for each page allocated
(vm-page-free-min = 20)
Only privileged tasks can run
(vm-page-free-reserved = 10)
Free Page List
ZK-0933U-AI

Detailed descriptions of the attributes are as follows:

- `vm-page-free-target`—Paging starts when the number of pages on the free list is less than this value (the default is 128 pages).
- `vm-page-free-min`—Specifies the threshold at which a page must be reclaimed for each page allocated (the default is 20 pages).
- `vm-page-free-swap`—Idle task swapping starts when the number of pages on the free list is less than this value for a period of time (the default is 74 pages).
- `vm-page-free-optimal`—Hard swapping starts when the number of pages on the free list is less than this value for five seconds (the default is 74 pages). The first processes to be swapped out include those with the lowest scheduling priority and those with the largest resident set size.
- `vm-page-free-hardswap`—Swapping stops when the number of pages on the free list is more than this value (the default is 1280 pages).
- `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 is 10 pages).

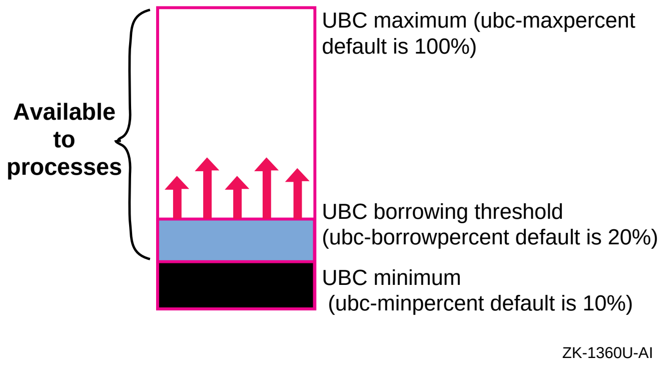
See Section 4.3.8 and Section 4.3.9 for information about paging and swapping operations.

4.3.7 Using Attributes to Control UBC Memory Allocation

Because the UBC shares with the virtual memory subsystem the physical pages that are not wired by the kernel, the allocation of memory to the UBC can affect file system performance and paging and swapping activity. The UBC is dynamic and consumes varying amounts of memory in order to respond to changing file system demands.

Figure 4–7 shows how memory is allocated to the UBC.

Figure 4–7: UBC Memory Allocation


Available to processes
UBC maximum (ubc-maxpercent default is 100%)
UBC borrowing threshold (ubc-borrowpercent default is 20%)
UBC minimum (ubc-minpercent default is 10%)
ZK-1360U-AI

The following attributes control the amount of memory available to the UBC:

- `ubc-minpercent` attribute

  Specifies the minimum percentage of memory that the UBC can utilize. The default is 10 percent.

- `ubc-maxpercent` attribute

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

- `ubc-borrowpercent` attribute

  Specifies the UBC borrowing threshold. The default is 20 percent. From the value of the `ubc-borrowpercent` attribute to the value of the `ubc-maxpercent` attribute, the UBC is only borrowing memory from the virtual memory subsystem. When paging starts, pages are first reclaimed from the UBC until the amount of memory allocated to the UBC reaches the value of the `ubc-borrowpercent` attribute.

4.3.8 Paging Operation

When the memory demand is high and the number of pages on the free page list reaches the value of the `vm-page-free-target` attribute, the virtual memory subsystem uses paging to replenish the free page list. The page reclamation code controls paging and swapping. The page-out daemon and task swapper daemon are extensions of the page reclamation code. See Section 4.3.6 for more information about the attributes that control paging and swapping.

The page reclamation code activates the page-stealer daemon, which first reclaims the pages that the UBC has borrowed from the virtual memory subsystem, until the size of the UBC reaches the borrowing threshold (the default is 20 percent). 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. Freeing borrowed UBC pages is a fast way to reclaim pages, because UBC pages are usually unmodified. See Section 4.3.7 for more information about UBC borrowed pages.

If freeing UBC borrowed memory does not sufficiently replenish the free list, a **pageout** occurs. The page-stealer daemon reclaims the oldest inactive and UBC LRU pages.

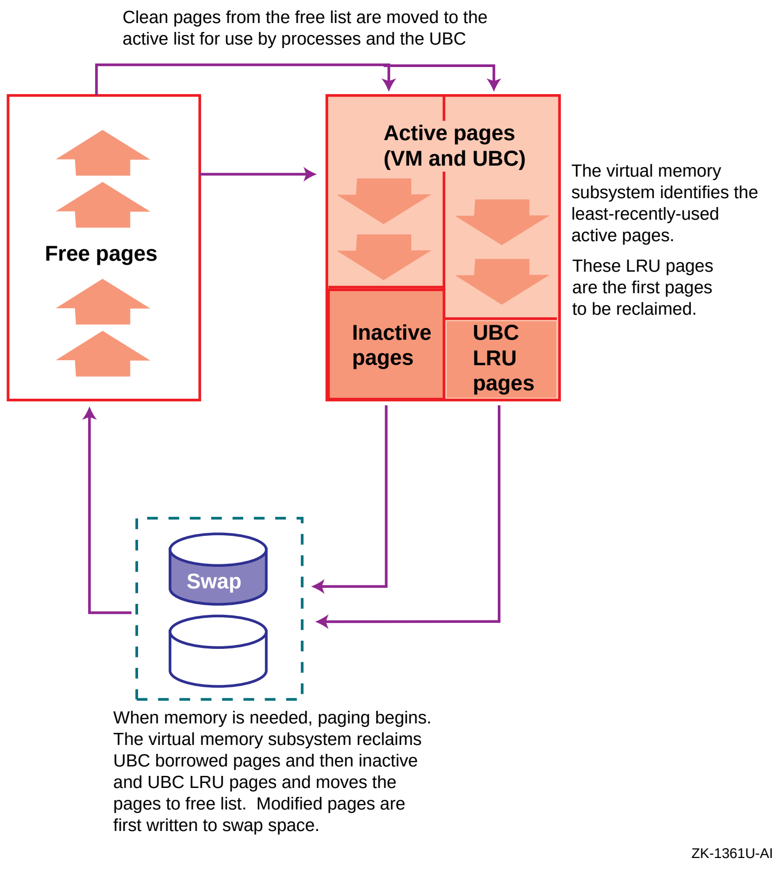
Paging becomes increasingly aggressive if the number of free pages continues to decrease. If the number of pages on the free page list falls below the value of the `vm-page-free-min` attribute (the default is 20 pages), a page must be reclaimed for each page allocated. To prevent deadlocks, if the number of pages on the free page list falls below the value of the `vm-page-free-reserved` attribute (the default is 10 pages), only privileged tasks can get memory until the free page list is replenished.

Paging stops when the number of pages on the free list reaches the value of the `vm-page-free-target` attribute.

If paging individual pages does not replenish the free list, swapping is used to free a large amount of memory. See Section 4.3.9 for more information.

Figure 4–8 shows the movement of pages during paging operations.

Figure 4–8: Paging Operation


Clean pages from the free list are moved to the active list for use by processes and the UBC
Free pages
Active pages (VM and UBC)
Inactive pages
UBC LRU pages
The virtual memory subsystem identifies the least-recently-used active pages.
These LRU pages are the first pages to be reclaimed.
Swap
When memory is needed, paging begins. The virtual memory subsystem reclaims UBC borrowed pages and then inactive and UBC LRU pages and moves the pages to free list. Modified pages are first written to swap space.
ZK-1361U-AI

4.3.9 Swapping Operation

If there is a high demand for memory, the virtual memory subsystem may be unable to replenish the free list by reclaiming pages. Swapping reduces the demand for physical memory by suspending processes, which dramatically increases the number of pages on the free list. To **swap out** a process, the task swapper suspends the process, writes its resident set to swap space, and moves the clean pages to the free list.

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 default is 74 pages). The task swapper suspends all tasks that have been idle for 30 seconds or more.

If the number of pages on the free list falls below the value of the `vm-page-free-optimal` attribute (the default is 74 pages) for more than five seconds, hard swapping begins. 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 reaches the value of the `vm-page-free-hardswap` attribute (the default is 1280).

A **swapin** occurs when the number of pages on the free list reaches the value of the `vm-page-free-optimal` attribute for a period of time. The task's working set is paged in from swap space and it can now execute. 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 outswapped. The default value is 1 second.

Swapping has a serious impact on system performance. You can modify the attributes described in Section 4.3.6 to control when swapping starts and stops.

Increasing the rate of swapping (swapping earlier during page reclamation) increases 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 degrades interactive response time. When an outswapped process is needed, it will have a long latency.

If you decrease the rate of swapping (swap later during page reclamation), you will improve interactive response time, but at the cost of throughput.

4.3.10 Using Swap Buffers

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 pageouts and for prewriting modified pages. See Section 4.7.15 and Section 4.7.16 for tuning information.

4.4 Understanding the Unified Buffer Cache

The DIGITAL UNIX operating system uses the Unified Buffer Cache (UBC) as a layer between the operating system and disk. The UBC holds actual file data, which includes reads and writes from conventional file activity and page faults from mapped file sections, and AdvFS metadata. The cache can improve I/O performance by decreasing the number of disk I/O operations.

The UBC shares with the virtual memory subsystem the physical pages that are not wired by the kernel. The maximum and minimum percentages of memory that the UBC can utilize are specified by the `ubc-maxpercent` attribute (the default is 100 percent) and the `ubc-minpercent` attribute (the default is 10 percent). In addition, the `ubc-borrowpercent` attribute specifies the percentage of memory allocated to the UBC above which the memory is only borrowed from the virtual memory subsystem. The default is 20 percent of physical memory. See Section 4.3.7 for more information.

The UBC is dynamic and consumes varying amounts of memory in order to respond to changing file system demands. For example, if file system activity is heavy, pages will be allocated to the UBC up to the value of the `ubc-maxpercent` attribute. 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. Figure 4–7 shows the allocation of physical memory to the UBC.

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-ubcbuffers` attribute specifies maximum file system device I/O queue depth for writes (that is, the number of UBC I/O requests that can be outstanding). See Section 4.7.17 for tuning information.

4.5 Understanding the Metadata Buffer Cache

The metadata buffer cache is part of kernel wired memory and is used to cache only UFS and CDFS metadata, which includes file header information, superblocks, inodes, indirect blocks, directory blocks, and cylinder group summaries. The DIGITAL UNIX operating system uses the metadata buffer cache as a layer between the operating system and disk. The cache can improve I/O performance by decreasing disk I/O operations.

The metadata buffer cache is configured at boot time and uses `bcopy` routines to move data in and out of memory. The size of the metadata buffer cache is specified by the value of the `bufcache` attribute. See Section 4.9 for tuning information.

4.6 Configuring Memory and Swap Space

The following sections describe how to configure memory and swap space, which includes the following tasks:

- Determining how much physical memory your system requires (Section 4.6.1)
- Determining how much swap space you need (Section 4.6.2)
- Choosing a swap space allocation mode (Section 4.6.3)

4.6.1 Determining Your Physical Memory Requirements

This section describes how to determine your system's memory requirements. The amount of memory installed in your system must be able to provide an acceptable level of user and application performance.

To determine your system's memory requirements, you must gather the following information:

- The amount of memory that will be wired
- The amount of memory that the virtual memory subsystem requires to cache the anonymous regions of process data
- The amount of memory that the UBC requires to cache file system data

See Section 4.6.2 for information about swap space requirements.

4.6.2 Configuring Swap Space

Your system's performance depends on the swap space configuration. DIGITAL recommends a minimum of 128 MB for swap space.

To calculate the swap space required by your system and workload, compare the total modifiable virtual address space (anonymous memory) required by your processes with the total amount of physical memory. Modifiable virtual address space holds data elements and structures that are modified during process execution, such as heap space, stack space, and data space.

To calculate swap space requirements if you are using immediate mode, total the anonymous memory requirements for all processes and then add 10 percent of that value. If you are using deferred mode, total the anonymous memory requirements for all processes and then divide by two.

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:

```
“lack of paging space”
“swap space below 10 percent free”
```

Use multiple disks for swap space. The page reclamation code uses a form of disk striping (known as swap space interleaving) so that pages can be written to the multiple disks. To optimize swap space, ensure that all your swap disks are configured when you boot the system, instead of adding swap space while the system is running.

Use the `swapon -s` command to display your swap space configuration. The first line displayed is the total allocated swap space. Use the `iostat` to display disk usage.

The following list describes how to configure swap space for high performance:

- Configure all of your swap space at boot time
- Use fast disks for swap space to decrease page fault latency
- Do not use busy disks for swap space
- Spread out your swap space across multiple disks (never put multiple swap partitions on the same disk)
- Spread out your swap disks across multiple I/O buses to prevent a single bus from becoming a bottleneck
- Use the Logical Storage Manager (LSM) to stripe your swap disks

See Chapter 5 for more information about configuring and tuning swap disks for high performance and availability.

4.6.3 Choosing a Swap Space Allocation Mode

There are two methods that you can use to allocate swap space. The methods differ in the point in time at which the virtual memory subsystem reserves swap space for a process. There is no performance benefit attached to either method; however, deferred mode is recommended for very-large memory/very-large database (VLM/VLDB) systems. The swap allocation methods are as follows:

- **Immediate mode**—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.

  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` or `sbrk` routines). When anonymous memory is allocated, the operating system reserves swap space for the memory. Usually, this results in an unnecessary amount of reserved swap space. Immediate mode requires more swap space than deferred mode, but it ensures that the swap space will be available to processes when it is needed.

- **Deferred mode**—Swap space is not reserved until the virtual memory subsystem needs to write a modified virtual page to swap space. Deferred mode is sometimes referred to as **lazy mode**.

  Deferred mode requires less swap space than immediate mode and causes the system to run faster because it requires less swap space bookkeeping. It postpones the reservation and allocation of swap space for anonymous memory until it is needed. However, because deferred mode does not reserve swap space in advance, the swap space may not be available when a task needs it, and the process may be killed asynchronously.

  You can enable the deferred swap space allocation mode by removing or moving the `/sbin/swapdefault` file.

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

4.7 Tuning Virtual Memory

The virtual memory subsystem is a primary source of performance problems. Performance may degrade if the virtual memory subsystem cannot keep up with the demand for memory and excessive paging and swapping occurs. A memory bottleneck may cause a disk I/O bottleneck, because excessive paging and swapping decreases performance and indicates that the natural working set size has exceeded the available memory. The virtual memory subsystem runs at a high priority when servicing page faults, which blocks the execution of other processes.

If you have excessive page-in and page-out activity from a swap partition, the system may have a high physical memory commitment ratio. Excessive paging also can increase the miss rate for the secondary cache, and may be indicated by the following output:

- The output of the `vmstat` shows a very low free page count or shows high page-in and page-out activity. See Section 2.4.2 for more information.
- The output of the `ps` command shows high task swapping activity. See Section 2.4.1 for more information.
- The output of the `iostat` command shows excessive swap disk I/O activity. See Section 2.5.1 for more information.

The tuning recommendations that will provide the best performance benefit involve the following two areas:

- System resource allocation
  - Increasing the available address space

    - Increasing the kernel resources available to processes
- Memory allocation and page reclamation
    - Modifying the percentage of memory allocated to the UBC
    - Changing the rate of swapping
    - Changing how the system prewrites modified inactive pages

Table 4–2 describes the primary tuning tasks guidelines and lists the performance benefits as well as tradeoffs.

Table 4–2: Primary Virtual Memory Tuning Guidelines

Action	Performance Benefit	Tradeoff
Reduce the number of processes running at the same time (Section 4.7.1)	Reduces demand for memory	None
Reduce the static size of the kernel (Section 4.7.2)	Reduces demand for memory	None
Increase the available address space (Section 4.7.3)	Improves performance for memory-intensive processes	Slightly increases the demand for memory
Increase the available system resources (Section 4.7.4)	Improves performance for memory-intensive processes	Increases wired memory
Increase the maximum number of memory-mapped files that are available to a process (Section 4.7.5)	Increases file mapping and improves performance for memory-intensive processes, such as Internet servers	Consumes memory
Increase the maximum number of virtual pages within a process' address space that can have individual protection attributes (Section 4.7.6)	Improves performance for memory-intensive processes and for Internet servers that maintain large tables or resident images	Consumes memory
Increase the size of a System V message and queue (Section 4.7.7)	Improves performance for memory-intensive processes	Consumes memory
Increase the maximum size of a single System V shared memory region (Section 4.7.8)	Improves performance for memory-intensive processes	Consumes memory
Increase the minimum size of a System V shared memory segment (Section 4.7.9)	Improves performance for VLM and VLDB systems	Consumes memory
Reduce process memory requirements (Section 4.7.10)	Reduces demand for memory	None

Table 4–2: Primary Virtual Memory Tuning Guidelines (cont.)

Action	Performance Benefit	Tradeoff
Reduce the amount of physical memory available to the UBC (Section 4.7.11)	Provides more memory resources to processes	May degrade file system performance
Increase the rate of swapping (Section 4.7.12)	Frees memory and increases throughput	Decreases interactive response performance
Decrease the rate of swapping (Section 4.7.12)	Improves interactive response performance	Decreases throughput
Increase the rate of dirty page prewriting (Section 4.7.13)	Prevents drastic performance degradation when memory is exhausted	Decreases peak workload performance
Decrease the rate of dirty page prewriting (Section 4.7.13)	Improves peak workload performance	May cause drastic performance degradation when memory is exhausted

If the previous tasks do not sufficiently improve performance, there are advanced tuning tasks that you can perform. The advanced tuning tasks include the following:

- Modify the sizes of the page-in and page-out clusters
- Modify the swap device I/O queue depth
- Modify the amount of memory the UBC uses to cache large files
- Increase the paging threshold
- Enable aggressive task swapping
- Decrease the size of the file system caches
- Reserve memory at boot time for shared memory

Table 4–3 describes the advanced tuning tasks guidelines and lists the performance benefits as well as tradeoffs.

Table 4–3: Advanced Virtual Memory Tuning Guidelines

Action	Performance Benefit	Tradeoff
Increase the size of the page-in and page-out clusters (Section 4.7.14)	Improves peak workload performance	Decreases total system workload performance
Decrease the size of the page-in and page-out clusters (Section 4.7.14)	Improves total system workload performance	Decreases peak workload performance
Increase the swap device I/O queue depth for pageins and swapouts (Section 4.7.15)	Increases overall system throughput	Consumes memory
Decrease the swap device I/O queue depth for pageins and swapouts (Section 4.7.15)	Improves the interactive response time and frees memory	Decreases system throughput
Increase the swap device I/O queue depth for pageouts (Section 4.7.16)	Frees memory and increases throughput	Decreases interactive response performance
Decrease the swap device I/O queue depth for pageouts (Section 4.7.16)	Improves interactive response time	Consumes memory
Increase the UBC write device queue depth (Section 4.7.17)	Increases overall file system throughput and frees memory	Decreases interactive response performance
Decrease the UBC write device queue depth (Section 4.7.17)	Improves interactive response time	Consumes memory
Increase the amount of UBC memory used to cache a large file (Section 4.7.18)	Improves large file performance	May allow a large file to consume all the pages on the free list
Decrease the amount of UBC memory used to cache a large file (Section 4.7.18)	Prevents a large file from consuming all the pages on the free list	May degrade large file performance
Increase the paging threshold (Section 4.7.19)	Maintains performance when free memory is exhausted	May waste memory
Enable aggressive swapping (Section 4.7.20)	Improves system throughput	Degrades interactive response performance
Decrease the size of the metadata buffer cache (Section 4.7.21)	Provides more memory resources to processes on large systems	May degrade UFS performance
Decrease the size of the namei cache (Section 4.7.22)	Decreases demand for memory	May slow lookup operations and degrade file system performance

Table 4–3: Advanced Virtual Memory Tuning Guidelines (cont.)

Action	Performance Benefit	Tradeoff
Decrease the amount of memory allocated to the AdvFS cache (Section 4.7.23)	Provides more memory resources to processes	May degrade AdvFS performance
Reserve physical memory for shared memory (Section 4.7.24)	Improves shared memory detach time	Decreases the memory available to the virtual memory subsystem and the UBC

The following sections describe these guidelines in detail.

4.7.1 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.

4.7.2 Reducing the Static Size of the Kernel

You can reduce the static size of the kernel by deconfiguring any unnecessary subsystems. Use the `setld` command to display the installed subsets and to delete subsets.

Use the `sysconfig` command to display the configured subsystems and to delete subsystems.

4.7.3 Increasing the Available Address Space

If your applications are memory-intensive, you may want to increase the available address space. Increasing the address space will cause only a small increase in the demand for memory. However, you may not want to increase the address space if your applications use many forked processes.

The following attributes determine the available address space for processes:

- `vm-maxvas`

  This attribute controls the maximum amount of virtual address space available to a process. The default value is 1 GB (1073741824). For Internet servers, you may want to increase this value to 10 GB.

- `per-proc-address-space` and `max-per-proc-address-size`

  These attributes control the maximum amount of user process address space, which is the maximum number of valid virtual regions. The default value for both attributes is 1 GB.

- `per-proc-stack-size` and `max-per-proc-stack-size`

  These attributes control the maximum size of a user process stack. The default value of the `per-proc-stack-size` attribute is 2097152 bytes. The default value of the `max-per-proc-stack-size` attribute is 33554432 bytes. You may need to increase these values if you receive `cannot grow stack` messages.

- `per-proc-data-size` and `max-per-proc-data-size`

  These attributes control the maximum size of a user process data segment. The default value of the `per-proc-data-size` attribute is 134217728 bytes. The default value of the `max-per-proc-data-size` is 1 GB.

You can use the `setrlimit` function to control the consumption of system resources by a parent process and its child processes. See `setrlimit`(2) for information.

4.7.4 Increasing the Available System Resources

If your applications are memory-intensive, you may want to increase the system resources that are available to processes. Be careful when increasing the system resources, because this will increase the amount of wired memory in the system.

The following attributes affect system resources:

- `maxusers`

  The `maxusers` attribute specifies the number of simultaneous users that a system can support without straining system resources. System algorithms use the `maxusers` attribute to size various system data structures, and to determine the amount of space allocated to system tables, such as the system process table, which is used to determine how many active processes can be running at one time.

  The default value assigned to the `maxusers` attribute depends on the size of your system. Increasing the value of the `maxusers` attribute allocates more system resources for use by the kernel. However, this also increases the amount of physical memory consumed by the kernel. Decreasing the value of the `maxusers` attribute reduces kernel memory usage, but allocates less system resources to processes.

  If your system experiences a lack of resources (for example, `Out of processes` messages), you can increase the value of the `maxusers` attribute to 512. A lack of resources may also be indicated by a `No more processes` error message. If you have sufficient memory on a heavily loaded system (for example, more than 96 MB), you can increase the value of the `maxusers` attribute to 1024.

- `task-max`

  The `task-max` attribute specifies the maximum number of tasks that can run simultaneously. The default value is 20 + 8 * `maxusers`.

- `thread-max`

  The `thread-max` attribute specifies the maximum number of threads. The default value is 2 * `task-max`.

- `max-proc-per-user`

  The `max-proc-per-user` attribute specifies the maximum number of processes that can be allocated at any one time to each user, except superuser. The default value of the `max-proc-per-user` attribute is 64.

  If your system experiences a lack of processes, you can increase the value of the `max-proc-per-user` attribute. The value must be more than the maximum number of processes that will be started by your system. If you have a Web server, these processes include CGI processes. If you plan to run more than 64 Web server daemons simultaneously, increase the attribute value to 512. On a very busy server with sufficient memory, you can use a higher value. Increasing this value can improve the performance of multiprocess Web servers.

- `max-threads-per-user`

  The `max-threads-per-user` attribute specifies the maximum number of threads that can be allocated at any one time to each user, except superuser. The default value is 256.

  If your system, especially a Web server, experiences a lack of threads, you can increase the value of the `max-threads-per-user` attribute. The value must be more than the maximum number of threads that will be started by your system. You can increase the value of the `max-threads-per-user` attribute to 512. On a very busy server with sufficient memory, you can use a higher value, such as 4096. Increasing this value can improve the performance of multithreaded Web servers.

You can use the `setrlimit` function to control the consumption of system resources by a parent process and its child processes. See `setrlimit`(2) for information.

4.7.5 Increasing the Number of Memory-Mapped Files

The `vm-mapentries` attribute specifies the maximum number of memory-mapped files in a user address. Each map entry describes one unique disjoint portion of a virtual address space. The default value is 200.

You may want to increase the value of the `vm-mapentries` attribute for VLM systems. Because Web servers map files into memory, for busy systems running multithreaded Web server software, you may want to increase the value to 20000. This will increase the limit on file mapping. This attribute affects all processes, and increasing its value will increase the demand for memory.

4.7.6 Increasing the Number of Pages With Individual Protections

The `vm-vpagemax` attribute specifies the maximum number of virtual pages within a process' address space that can be given individual protection attributes. These protection attributes differ from the protection attributes associated with the other pages in the address space.

Changing the protection attributes of a single page within a virtual memory region causes all pages within that region to be treated as though they had individual protection attributes. For example, each thread of a multithreaded task has a user stack in the stack region for the process in which it runs. Because multithreaded tasks have guard pages (that is, pages that do not have read/write access) inserted between the user stacks for the threads, all pages in the stack region for the process are treated as though they have individual protection attributes.

The default value of the `vm-vpagemax` attribute is determined by dividing the value of the `vm-maxvas` attribute (the address space size in bytes) by 8192. If a stack region for a multithreaded task exceeds 16 KB pages, you may want to increase the value of the `vm-vpagemax` attribute. For example, if the value of the `vm-maxvas` attribute is 1 GB (the default), set the value of `vm-vpagemax` to 131072 pages (1073741824/8192=131072). This value improves the efficiency of Web servers that maintain large tables or resident images.

You may want to increase the value of the `vm-vpagemax` attribute for VLM systems. However, this attribute affects all processes, and increasing its value will increase the demand for memory.

4.7.7 Increasing the Size of a System V Message and Queue

If your applications are memory-intensive or you have a VLM system, you may want to increase the value of the `msg-max` attribute. This attribute specifies the maximum size of a single System V message. However, increasing the value of this attribute will increase the demand for memory. The default value is 8192 bytes (1 page).

In addition, you may want to increase the value of the `msg-tql` attribute. This attribute specifies the maximum number of messages that can be queued to a single System V message queue at one time. However, increasing the value of this attribute will increase the demand for memory. The default value is 40.

4.7.8 Increasing the Size of a System V Shared Memory Region

If your applications are memory-intensive or you have a VLM system, you may want to increase the value of the `shm-max` attribute. This attribute specifies the maximum size of a single System V shared memory region. However, increasing the value of this attribute will increase the demand for memory. The default value is 4194304 bytes (512 pages).

In addition, you may want to increase the value of the `shm-seg` attribute. This attribute specifies the maximum number of System V shared memory regions that can be attached to a single process at any point in time. However, increasing the value of this attribute will increase the demand for memory. The default value is 32.

4.7.9 Increasing the Minimum Size of a System V Shared Memory Segment

If your applications are memory-intensive, you may want to increase the value of the `ssm-threshold` attribute. Page table sharing occurs when the size of a System V shared memory segment reaches the value specified by this attribute. However, increasing the value of this attribute will increase the demand for memory.

4.7.10 Reducing Application Memory Requirements

You may want to reduce your applications' use of memory to free memory for other purposes. Follow these coding considerations to reduce your applications' 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.
- 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` flag. The `-xtaso` flag is supported by all versions of the C compiler (`-newc`, `-migrate`, and `-oldc` versions). To use the `-xtaso` flag, 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 more information on process memory allocation.

4.7.11 Reducing the Memory Available to the UBC

You may be able to improve performance by reducing the maximum percentage of memory available for the UBC. If you decrease the maximum size of the UBC, you increase the amount of memory available to the virtual memory subsystem, which may reduce the paging and swapping rate. However, reducing the memory allocated to the UBC may adversely affect I/O performance because the UBC will hold less file system data, which results in more disk I/O operations. Therefore, do not significantly decrease the maximum size of the UBC.

The maximum amount of memory that can be allocated to the UBC is specified by the `ubc-maxpercent` attribute. The default is 100 percent. The minimum amount of memory that can be allocated to the UBC is specified by the `ubc-minpercent` attribute. The default is 10 percent. If you have an Internet server, use these default values.

If the page-out rate is high and you are not using the file system heavily, decreasing the value of the `ubc-maxpercent` attribute may reduce the rate of paging and swapping. Start with the default value of 100 percent and decrease the value in increments of 10. If the values of the `ubc-maxpercent` and `ubc-minpercent` attributes are close together, you may seriously degrade I/O performance or cause the system to page excessively.

Use the `vmstat` command to determine whether the system is paging excessively. Using `dbx`, periodically examine the `vpf_pgiowrites` and `vpf_ubcalloc` fields of the `vm_perfsum` kernel structure. The page-out rate may shrink if pageouts greatly exceed UBC allocations.

You also may be able to prevent paging by increasing the percentage of memory that the UBC borrows from the virtual memory subsystem. To do this, decrease the value of the `ubc-borrowpercent` attribute. Decreasing the value of the `ubc-borrowpercent` attribute allows less memory to remain in the UBC when page reclamation begins. This can reduce the UBC effectiveness, but may improve the system response time when a low-memory condition occurs. The value of the `ubc-borrowpercent` attribute can range from 0 to 100. The default value is 20 percent.

4.7.12 Changing the Rate of Swapping

Swapping has a drastic impact on system performance. You can modify attributes to control when swapping begins and ends. Increasing the rate of swapping (swapping earlier during page reclamation), 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 will degrade interactive response time.

In contrast, if you decrease the rate of swapping (swap later during page reclamation), you will improve interactive response time, but at the cost of throughput.

To increase the rate of swapping, increase the value of the `vm-page-free-optimal` attribute (the default is 74 pages). Increase the value only by 2 pages at a time. Do not specify a value that is more than the value of the `vm-page-free-target` attribute (the default is 128).

To decrease the rate of swapping, 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-page-free-min` attribute (the default is 20).

4.7.13 Controlling Dirty Page Prewriting

The virtual memory subsystem attempts to prevent a memory shortage by prewriting modified pages to swap space. When the virtual memory subsystem anticipates that the pages on the free list will soon be depleted, it prewrites to swap space the oldest modified (dirty) pages on the inactive list. To reclaim a page that has been prewritten, the virtual memory subsystem only needs to validate the page.

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

You can control the rate of dirty page prewriting by modifying the values of the `vm-page-prewrite-target` attribute and the `vm-ubcdirtypercent` attribute.

The `vm-page-prewrite-target` attribute specifies the number of virtual memory pages that the subsystem will prewrite and keep clean. The default value is 256 pages. To increase the rate of virtual memory dirty page prewriting, increase the value of the `vm-page-prewrite-target` attribute from the default value (256) by increments of 64 pages.

The `vm-ubcdirtypercent` attribute specifies the percentage of UBC LRU pages that can be modified before the virtual memory subsystem prewrites the dirty UBC LRU pages. The default value 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 increase the rate of UBC LRU dirty page prewriting, decrease the value of the `vm-ubcdirtypercent` attribute by increments of 1 percent.

In addition, you may want to minimize the impact of I/O spikes caused by the `sync` function when prewriting UBC LRU dirty pages. The value of the `ubc-maxdirtywrites` attribute specifies the maximum number of disk writes that the kernel can perform each second. The default value of the `ubc-maxdirtywrites` attribute is 5 I/O operations per second.

To minimize the impact of `sync` (steady state flushes) when prewriting dirty UBC LRU pages, increase the value of the `ubc-maxdirtywrites` attribute.

4.7.14 Modifying 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-max-rdpgio-kluster` attribute specifies the maximum size of an anonymous page-in cluster. The default value is 16 KB (2 pages). If you increase the value of this 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 the total system workload performance.

Decreasing the value of the `vm-max-rdpgio-kluster` attribute will conserve memory and increase the total system workload performance, but will increase paging and decrease the peak workload performance.

The `vm-max-wrpgio-kluster` attribute specifies the maximum size of an anonymous page-out cluster. The default value is 32 KB (4 pages). Increasing the value of this attribute improves the peak workload performance and conserves memory, but causes more pageins and decreases the total system workload performance.

Decreasing the value of the `vm-max-wrpgio-kluster` attribute improves the total system workload performance and decreases the number of pageins, but decreases the peak workload performance and consumes more memory.

4.7.15 Modifying the Swap I/O Queue Depth for Pageins and Swapouts

Synchronous swap buffers are used for page-in page faults and for swapouts. The `vm-syncswapbuffers` attribute specifies the maximum swap device I/O queue depth for pageins and swapouts.

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

Increasing the swap device I/O queue depth increases overall system throughput, but consumes memory.

Decreasing the swap device I/O queue depth decreases memory demands and improves interactive response time, but decreases overall system throughput.

4.7.16 Modifying the Swap I/O Queue Depth for Pageouts

Asynchronous swap buffers are used for asynchronous pageouts and for prewriting modified pages. The `vm-asyncswapbuffers` attribute controls the maximum depth of the swap device I/O queue for pageouts.

The value of the `vm-asyncswapbuffers` attribute should be the approximate number of I/O transfers that a swap device can handle at one time. The default value is 4.

Increasing the queue depth will free memory and increase the overall system throughput.

Decreasing the queue depth will use more memory, but will improve the interactive response time.

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.

4.7.17 Modifying the UBC Write Device Queue Depth

The UBC uses a buffer to facilitate the movement of data between memory and disk. The `vm-ubcbuffers` attribute specifies the maximum file system device I/O queue depth for writes. The default value is 256.

Increasing the UBC write device queue depth frees memory and increases the overall file system throughput.

Decreasing the UBC write device queue depth increases memory demands, but improves the interactive response time.

4.7.18 Controlling Large File Caching

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-ubcseqpercent` attribute specifies the maximum amount of memory allocated to the UBC that can be used to cache a file. The default value is 10 percent of memory allocated to the UBC.

The `vm-ubcseqstartpercent` attribute 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. The default is 50 percent of physical memory.

Increasing the value of the `vm-ubcseqpercent` attribute will improve the performance of a large single file, but decrease the remaining amount of memory.

Decreasing the value of the `vm-ubcseqpercent` attribute will increase the available memory, but will degrade the performance of a large single file.

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 percentage of memory. That is, the value of the `ubc-maxpercent` attribute (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 lower the `vm-ubcseqstartpercent` value 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.

4.7.19 Increasing the Paging Threshold

The `vm-page-free-target` attribute specifies the minimum number of pages on the free list before paging starts. The default value is 128 pages.

Increasing the value of the `vm-page-free-target` attribute will increase the paging activity but may improve performance when free memory is exhausted. If you increase the value, start at the default value (128 pages or 1 MB) and then double the value. Do not specify a value above 1025 pages or 8 MB. A high value can waste memory.

Do not decrease the value of the `vm-page-free-target` attribute unless you have a lot of memory or you experience a serious performance degradation when free memory is exhausted.

4.7.20 Enabling Aggressive Task Swapping

You can enable the `vm-aggressive` attribute (set the value to 1) to allow the virtual memory subsystem to aggressively swap out processes when memory is needed. This improves system throughput, but degrades the interactive response performance.

By default, the `vm-aggressive` attribute is disabled (set to 0), which results in less aggressive swapping. In this case, processes are swapped in at a faster rate than if aggressive swapping is enabled.

4.7.21 Decreasing the Size of the Metadata Buffer Cache

The metadata buffer cache contains recently accessed UFS and CDFS metadata. On large-memory systems with a high cache hit rate, you may want to decrease the size of the metadata buffer cache. This will increase the amount of memory that is available to the virtual memory subsystem. However, decreasing the size of the cache may degrade UFS performance.

The `bufcache` attribute specifies the percentage of physical memory that the kernel wires for the metadata buffer cache. 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 systems that use only AdvFS, set the value of the `bufcache` attribute to 1 percent.

4.7.22 Decreasing the Size of the namei Cache

The namei cache is used by all file systems to map file pathnames to inodes. Use `dbx` to monitor the cache by examining the `nchstats` structure.

To free memory resources, decrease the number of elements in the namei cache by decreasing the value of the `name-cache-size` attribute. The default values are 2*`nvnode`*11/10 (for 32-MB or larger systems) and 150 (for 24-MB systems). The maximum value is 2*`max-vnodes`*11/10.

4.7.23 Decreasing the Size of the AdvFS Buffer Cache

To free memory resources, you may want to decrease the percentage of physical memory allocated to the AdvFS buffer cache.

The `AdvfsCacheMaxPercent` attribute determines the maximum amount of physical memory that can be used for the AdvFS buffer cache. The default is 7 percent of memory. However, decreasing the size of the AdvFS buffer cache may adversely affect AdvFS I/O performance.

4.7.24 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. Granularity hints allow the translation lookaside buffer to map more than a single page and enable shared page table entry functionality, which will cause fewer buffer misses.

On typical database servers, using granularity hints provides a 2 to 4 percent run-time performance gain that reduces the shared memory detach time. In most cases, use the Segmented Shared Memory (SSM) functionality (the default) instead of the granularity hints functionality.

To enable granularity hints, you must specify a value for the gh-chunks attribute. 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 4.7.24.1 and Section 4.7.24.2 describe how to enable granularity hints.

4.7.24.1 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 reserving memory decreases the memory available to the virtual memory subsystem and the UBC.

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:

```
# dbx -k /vmunix /dev/mem

(dbx) px &gh_free_counts
0xfffffc0000681748
(dbx) 0xfffffc0000681748/4X
fffffc0000681748:  0000000000000402 0000000000000004
fffffc0000681758:  0000000000000000 0000000000000002
(dbx)
```

The 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 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.

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.

4.7.24.2 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 Level 3 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:

```
# dbx -k /vmunix /dev/mem

(dbx) p *(vm_granhint_stats *)&gh_stats_store
 struct {
     total_mappers = 21
     shared_mappers = 21
     unshared_mappers = 0
     total_unmappers = 21
     shared_unmappers = 21
     unshared_unmappers = 0
     unaligned_mappers = 0
     access_violations = 0
     unaligned_size_requests = 0
     unaligned_attachers = 0
     wired_bypass = 0
     wired_returns = 0
 }
 (dbx)
```

For the best performance, the shared_mappers kernel variable should be equal to the number of shared memory segments, and the unshared_mappers, unaligned_attachers, and unaligned_size_requests variables should be 0 (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-page-lock-count attribute. The default value is 64.

4.8 Tuning the UBC

The UBC and the virtual memory subsystem compete for the physical memory that is not wired by the kernel. You may be able to improve file system performance by tuning the UBC. However, increasing the amount of memory available to the UBC will affect the virtual memory subsystem and may increase the rate of paging and swapping.

The amount of memory allocated to the UBC is determined by the ubc-maxpercent, ubc-minpercent, and ubc-borrowpercent attributes. You may be able to improve performance by modifying the value of these attributes, which are described in Section 4.4.

The following 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.

The UBC is flushed by the `update` daemon. You can monitor the UBC usage lookup hit ratio by using `dbx`. You can view UBC statistics by using `dbx` and checking the `vm_perfsum` structure. You can also monitor the UBC by using `dbx -k` and examining the `ufs_getapage_stats` structure. See Chapter 2 for information about monitoring the UBC.

You can improve UBC performance by following the guidelines described in Table 4–4. You can also improve file system performance by following the guidelines described in Chapter 5.

Table 4–4: Guidelines for Tuning the UBC

Action	Performance Benefit	Tradeoff
Increase the memory allocated to the UBC (Section 4.8.1)	Improves file system performance	May cause excessive paging and swapping
Decrease the amount of memory borrowed by the UBC (Section 4.8.2)	Improves file system performance	Decreases the memory available for processes and may decrease system response time
Increase the minimum size of the UBC (Section 4.8.3)	Improves file system performance	Decreases the memory available for processes
Modify the application to use `mmap` (Section 4.8.4)	Decreases memory requirements	None
Increase the UBC write device queue depth (Section 4.7.17)	Increases overall file system throughput and frees memory	Decreases interactive response performance
Decrease the UBC write device queue depth (Section 4.7.17)	Improves interactive response time	Consumes memory

The following sections describe these guidelines in detail.

4.8.1 Increasing the Maximum Size of the UBC

If there is an insufficient amount of memory allocated to the UBC, I/O performance may be degraded. If you allocate more memory to the UBC, you will improve the chance that data will be found in the cache. By preventing the system from having to copy data from a disk, you may improve I/O performance. However, allocating more memory to the UBC may cause excessive paging and swapping.

To increase the maximum amount of memory allocated to the UBC, you can increase the value of the `ubc-maxpercent` attribute. The default value is 100 percent. However, the performance of an application that generates a lot of random I/O will not be improved by enlarging the UBC because the next access location for random I/O cannot be predetermined. See Section 4.3.7 for information about UBC memory allocation.

4.8.2 Decreasing the Amount of Borrowed Memory

If `vmstat` output shows excessive paging but few or no pageouts, you may want to increase the value of the `ubc-borrowpercent` attribute. This situation can occur on low-memory systems (24-MB systems) because they reclaim UBC pages more aggressively than systems with more memory.

The UBC borrows all physical memory above the value of the `ubc-borrowpercent` attribute and up to the value of the `ubc-maxpercent` attribute. Increasing the value of the `ubc-borrowpercent` attribute allows more memory to remain in the UBC when page reclamation begins. This can increase the UBC cache effectiveness, but may degrade system response time when a low-memory condition occurs. The value of the `ubc-borrowpercent` attribute can range from 0 to 100. The default value is 20 percent. See Section 4.3.7 for information about UBC memory allocation.

4.8.3 Increasing the Minimum Size of the UBC

Increasing the value of the `ubc-minpercent` attribute will prevent large programs from completely filling the UBC. For I/O servers, you may want to raise the value of the `ubc-minpercent` attribute to ensure that memory is available for the UBC. The default value is 10 percent.

To ensure that the value of the `ubc-minpercent` is appropriate, use the `vmstat` command to examine the page-out rate.

If the values of the `ubc-maxpercent` and `ubc–minpercent` attributes are close together, you may degrade I/O performance or cause the system to page excessively. See Section 4.3.7 for information about UBC memory allocation.

4.8.4 Using mmap in Your Applications

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.

4.9 Tuning the Metadata Buffer Cache

A portion of physical memory is wired for use by the metadata buffer cache, which is the traditional BSD buffer cache. The file system code that deals with UFS metadata, which includes directories, indirect blocks, and inodes, uses this cache.

You may be able to improve UFS performance by following the guidelines described in Table 4–5.

Table 4–5: Guidelines for Tuning the Metadata Buffer Cache

Action	Performance Benefit	Tradeoff
Increase the memory allocated to the metadata buffer cache (Section 4.9.1)	Improves UFS performance	Reduces the memory available to the virtual memory subsystem and the UBC
Increase the size of the hash chain table (Section 4.9.2)	Improves lookup speed	Consumes memory

The following sections describe these guidelines in detail.

4.9.1 Increasing the Size of the Metadata Buffer Cache

The `bufcache` attribute specifies the size of the kernel's metadata buffer cache as a percentage of physical memory. The default is 3 percent.

You may want to increase the size of the metadata buffer cache if you have a high cache miss rate (low hit rate). In general, you do not have to increase the cache size. Never increase the value of the `bufcache` to more than 10 percent.

To determine whether to increase the size of the metadata buffer cache, use `dbx` to examine the `bio_stats` structure. The miss rate (block misses divided by the sum of the block misses and block hits) should not be more than 3 percent.

Allocating additional memory to the metadata buffer cache reduces the amount of memory available to the virtual memory subsystem and the

UBC. In general, you do not have to increase the value of the `bufcache` attribute.

4.9.2 Increasing the Size of the Hash Chain Table

The hash chain table for the metadata buffer cache stores the heads of the hashed buffer queues. Increasing the size of the hash chain table spreads out the buffers and may reduce linear searches, which improves lookup speeds.

The `buffer-hash-size` attribute specifies the size of the hash chain table for the metadata buffer cache. The default hash chain table size is 512 slots.

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 `dbx` to examine the value of `nbuf`, 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.

5 Configuring and Tuning Storage Subsystems

A storage subsystem consists of software (operating system or layered product) and hardware (including host bus adapters, cables, and disks). Your 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.

To configure a storage subsystem that will meet your performance and availability needs, you must first understand the I/O requirements of the users and applications and how they perform disk I/O, as described in Chapter 1. After you configure your storage subsystem, you may be able to tune the subsystem to improve performance.

This chapter describes the features of different storage subsystems and provides guidelines for configuring and tuning the subsystems.

Many of the tuning tasks described in this chapter require you to modify system attributes. See Section 2.11 for more information about attributes.

5.1 Understanding Storage Subsystems

Disk I/O operations are significantly slower than data transfers involving the CPU or memory caches. Because disks are used for data storage and for virtual memory swap space, an incorrectly configured or tuned storage subsystem can degrade overall system performance.

Disk I/O performance can be affected by the following variables:

- Workload characteristics

  Performance depends on how your users and applications perform disk I/O. For example, a workload can involve primarily read or write I/O operations. In addition, some workloads require low latency and high throughput, while others require a fast data transfer rate (high bandwidth).

  Low latency is important for multiple small data transfers and also for workstation, timesharing, and server environments. High bandwidth is important for systems that perform large sequential data transfers, such as database servers. See Chapter 1 for more information about characterizing your disk I/O.

- Performance capacity of the hardware

  DIGITAL recommends that you use hardware with the best performance features. For example, disks with a high rate of revolutions per minute (RPM) provide the best overall performance. Wide disks, which support 16-bit transfers, have twice the bandwidth of narrow (8 bit) disks and can improve performance for large data transfers. High-performance host bus adapters, such as fast wide differential (FWD) adapters provide low CPU overhead and high bandwidth. In addition, a write-back cache decreases the latency of small writes and can improve throughput.

- Memory allocation to the UBC

  The Unified Buffer Cache (UBC) is allocated a portion of physical memory and caches actual file system data for reads and writes, Advanced File System (AdvFS) metadata, and Memory File System (MFS) data. The UBC decreases the number of disk operations for file systems by serving as a layer between the disk and the operating system. The metadata buffer cache and the AdvFS buffer cache are also allocated a percentage of physical memory.

- Kernel variable values

  Disk I/O performance depends on kernel variable values that are appropriate for your workload and configuration. You may need to modify the default values to obtain optimal system performance, as described in this manual.

- Mirrored disk configuration

  Mirrored data across different disks improves the performance of read operations and provides high data availability. However, because data must be written to two separate locations, mirroring degrades disk write performance.

- Striped disk configuration

  Striping data across multiple disks distributes the I/O load and enables parallel I/O streams to operate concurrently on different devices, which improves disk I/O performance for some workloads.

- Hardware RAID subsystem configuration

  Hardware RAID subsystems relieve the CPU of disk management overhead and support write-back caches, which can improve disk I/O performance for some workloads.

- File system configuration

File systems, including UFS and AdvFS, are used to organize and manage files. AdvFS provides you with fast file system recovery, improved performance for sequential and large I/O operations, and disk defragmentation features.

- Raw I/O

  For some workloads, raw I/O (I/O to a disk that does not contain a file system) may have better performance than file system I/O because it bypasses buffers and caches.

To choose a storage subsystem that will meet the needs of your users and applications, you must understand the benefits and tradeoffs of the various disk and file management options, as described in Section 5.2.

5.2 Choosing How to Manage Disks and Files

DIGITAL UNIX supports a number of methods that you can use to manage the physical disks and files 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. A disk can consist of one to eight partitions that have a fixed size; these partitions cannot overlap.

Each disk type has a default partition scheme. The `disktab` database file lists the default disk partition sizes. The partition size determines the amount of data it can hold. 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. You must be sure that the data will fit into the new partition.

An alternative method to managing disks with static disk partitions is to use the Logical Storage Manager (LSM) to set up a shared storage pool that consists of multiple disks. You can then create virtual disks from this pool of storage, according to your performance and capacity needs. 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.

Hardware RAID subsystems provide another method of handling storage. These subsystems use intelligent controllers to provide high-performance and high-availability RAID functionality, allow you to increase your storage capacity, and support write-back caches. RAID controllers allow you to combine several disks into a single storage set that the system sees as a single unit.

You can choose to manage your file systems by using AdvFS. 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. This decoupled file system structure enables you to manage the physical storage layer apart from the directory hierarchy layer. This means that you can 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 end users.

You can use different configurations in your environment. For example, you can create static partitions on some disks and use the remaining disks in LSM volumes. You can also combine products in the same configuration. For example, you can configure AdvFS file domains on top of LSM volumes, or configure LSM volumes on top of RAID storage sets.

The following sections describe the features of the different disk and file system management options.

5.2.1 Understanding RAID Levels and Products

RAID (redundant array of independent disks) technology can provide both high disk I/O performance and high data availability. The DIGITAL UNIX operating system provides RAID functionality by using the Logical Storage Manager (LSM) product. DIGITAL UNIX also supports hardware RAID subsystems, which provide RAID functionality by using intelligent controllers, caches, and software.

There are four primary RAID levels:

- RAID 0—Also known as disk striping, RAID 0 divides data into blocks (sometimes called chunks or stripes) and distributes the blocks across multiple disks in a array. Striping enables parallel I/O streams to operate concurrently on different devices. I/O operations can be handled simultaneously by multiple devices, which balances the I/O load and improves performance.

  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 disks highly available, you can mirror the disks.

- RAID 1—Also known as disk mirroring, RAID 1 provides high data availability by maintaining identical copies of data on different disks in an array. RAID 1 also improves the disk read performance, because data can be read from two different locations. However, RAID 1 can decrease disk write performance, because data must be written to two different locations.
- 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. RAID 3 provides a high data transfer rate and increases bandwidth, but it provides no improvement in throughput (the I/O transaction rate).

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

  RAID 3 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—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.

  RAID 5 can improve throughput, especially for large file I/O operations, multiple small data transfers, and I/O read operations. However, it is not suited to write-intensive applications.

  RAID 5 provides 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.

To address your performance and availability needs, you can combine some RAID levels. For example, you can combine RAID 0 with RAID 1 to mirror striped disks for high availability and high performance.

In addition, some DIGITAL hardware RAID subsystems support adaptive RAID 3/5 (also called dynamic parity RAID), which improves disk I/O performance for a wide variety of applications by dynamically adjusting, according to workload needs, between data transfer-intensive algorithms and I/O operation-intensive algorithms.

Table 5–1 compares the performance and availability features for the different RAID levels.

Table 5–1: RAID Level Performance and Availability Features

RAID Level	Performance Impact	Availability Impact
RAID 0	Balances I/O load and improves reads and writes	Lower than single disk
RAID 1	Improves reads, may degrade writes	Highest
RAID 0+1	Balances I/O load, improves reads, may degrade writes	Highest
RAID 3	Improves bandwidth, performance may degrade if multiple disks fail	Higher than single disk
RAID 5	Improves throughput, performance may degrade if multiple disks fail	Higher than single disk
Adaptive RAID 3/5	Improves bandwidth and throughput, performance may degrade if multiple disks fail	Higher than single disk

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).

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

- Not all RAID products support all RAID levels.

  For example, LSM currently supports only RAID 0 (striping) and RAID 1 (mirroring), and only high-performance RAID controllers support adaptive RAID 3/5.

- RAID products provide different performance benefits.

  For example, hardware RAID subsystems support write-back caches and other performance-enhancing features and also 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.

There are advantages to each RAID product, and which one you choose depends on your workload requirements and other factors. The following sections describe the features of the different RAID subsystems and LSM.

5.2.1.1 Hardware RAID Subsystem Features

Hardware RAID subsystems use a combination of hardware (RAID controllers, caches, and host bus adapters) and software to provide high disk I/O performance and high data availability. A hardware RAID subsystem is sometimes called **hardware RAID**.

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 your system. In a typical storage configuration, you use a SCSI bus connected to an I/O bus slot to attach disks to a system. However, you can connect only a limited number of disks on a SCSI bus, and systems have limited I/O bus slots. Hardware RAID subsystems contain internal SCSI buses and host bus adapters, which enable you to connect multiple SCSI buses and multiple disks to a system by using only one I/O bus slot.
- Read cache

  A read cache can improve 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 (as a standard or an optional feature) a nonvolatile write-back cache, 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 Web server performance because writes appear to be executed immediately.

A write-back cache must be battery-backed to protect against data loss and corruption.

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. 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.

- RAID 0 (disk striping) support
- RAID 1 (disk mirroring) support
- Parity RAID support

  Hardware RAID subsystems provide various levels of parity RAID support (RAID 3, RAID 5, or adaptive RAID 3/5) for high performance and high availability.
- Hot component swapping and 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.
- Non-RAID disk array capability or "just a bunch of disks" (JBOD)
- Graphical user interface (GUI) for easy management and monitoring
- The `volstat` command, which provides detailed LSM performance information

There are various hardware RAID subsystems, including backplane RAID array subsystems and high-performance standalone RAID array subsystems, which provide different degrees of performance and availability at various costs. The features of these two subsystems are as follows:

- Backplane RAID array subsystems

  These entry-level subsystems, such as the RAID Array 230 subsystem, provide a low-cost hardware RAID solution. A backplane RAID array 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 are designed for small and midsize departments and workgroups, and provide RAID functionality (0, 1, 0+1, and 5) and an optional write-back cache.
- Standalone RAID array subsystems

  These subsystems, such as the RAID Array 450 subsystem, provide high availability and the highest performance of any RAID subsystem. A standalone RAID array subsystem uses a high-performance controller, such as the HSZ controller. The controller connects to the system through a FWD SCSI bus and a high-performance host bus adapter, such as a KZPSA adapter, installed in an I/O bus slot.

Standalone RAID array subsystems are designed for client/server, data center, and medium to large departmental environments. They provide RAID functionality (0, 1, 0+1, and adaptive RAID 3/5), dual-redundant controller support, scalability, storage set partitioning, and a standard write-back cache.

See Section 5.5 for information on configuring hardware RAID subsystems.

5.2.1.2 LSM Features

Logical Storage Manager (LSM) can improve disk I/O performance, provide high data availability, and help you to manage your storage more efficiently. All DIGITAL UNIX systems can use the basic LSM functions, but advanced disk management functions require a separate LSM license. When LSM is used to stripe or mirror disks, it is sometimes referred to as **software RAID**.

LSM allows you to organize a shared storage pool into volumes, which are used in the same way as disk partitions, except that I/O directed to a volume can span disks. You can create a UFS file system or an AdvFS file domain on a volume, or you can use a volume as a raw device. You can also create LSM volumes on top of RAID storage sets.

LSM supports the following disk management features:

- Pool of storage
- Load balancing by transparently moving data across disks
- RAID 0 (disk striping) support (license necessary)
- RAID 1 (disk mirroring) support (license necessary)
- Disk concatenation (creating a large volume from multiple disks)
- Graphical user interface (GUI) for easy disk management and detailed performance information (license necessary)

LSM provides more cost-effective RAID functionality than a hardware RAID subsystem. In addition, LSM configurations are less complex than hardware RAID configurations. To obtain the performance benefits of both LSM and hardware RAID, you can create LSM volumes on top of RAID storage sets.

LSM is especially suited for systems with large numbers of disks. For these systems, you may want to use LSM to manage your disks and AdvFS to manage your files. That is, you can organize your disks into LSM volumes and then use those volumes to create AdvFS file domains.

5.2.2 Understanding AdvFS

Advanced File System (AdvFS) is a DIGITAL UNIX file system option that provides many file management and performance features. You can use AdvFS instead of UFS to organize and manage your files.

The AdvFS Utilities product, which is licensed separately from the DIGITAL UNIX operating system, extends the capabilities of the AdvFS file system. An AdvFS file domain can consist of multiple volumes, which can be UNIX block devices (entire disks), disk partitions, LSM logical volumes, or RAID storage sets. AdvFS filesets can span all the volumes in the file domain.

AdvFS provides the following file management features:

- Fast file system recovery

  Rebooting after a system interruption is extremely fast. 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.

- 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.

- Online file system management
- File domain defragmentation
- Support for large files and file systems
- User quotas

AdvFS utilities provide the following features:

- Pool of storage that allows you to add, remove, and back up disks without disrupting users or applications.
- Disk spanning filesets
- Ability to recover deleted files

  Users can retrieve their own unintentionally deleted files from predefined trashcan directories or from clone filesets, without assistance from system administrators.

- I/O load balancing across disks
- Online fileset resizing

- Online file migration across disks
- File-level striping

  File-level striping may improve I/O bandwidth (transfer rates) by distributing file data across multiple disk volumes.
- Graphical user interface (GUI) that simplifies disk and file system administration, provides status, and alerts you to potential problems

See Section 5.6 for information about AdvFS configuration and tuning guidelines.

5.3 General Disk Storage Guidelines

There are some general guidelines for configuring and tuning storage subsystems. These guidelines are applicable to most configurations and will help you to get the best disk I/O performance, regardless of whether you are using static partitions, raw devices, LSM, hardware RAID subsystems, AdvFS, or UFS.

These guidelines fall into three categories:

- Using high-performance hardware (see Table 5–2)
- Distributing the disk I/O load (see Table 5–3)
- General file system tuning (see Table 5–4)

The following sections describe these guidelines in detail.

5.3.1 High-Performance Hardware Guidelines

Using high-performance hardware will provide the best disk I/O performance, regardless of your storage configuration. Table 5–2 describes the guidelines for hardware configurations and lists the performance benefits as well as the tradeoffs.

Table 5–2: Guidelines for High-Performance Hardware Configurations

Hardware	Performance Benefit	Tradeoff
Fast (high RPM) disks (Section 5.3.1.1)	Improve disk access time and sequential data transfer performance	Cost
Disks with small platter sizes (Section 5.3.1.2)	Improve seek times for applications that perform many small I/O operations	No benefit for large sequential data transfers
Wide disks (Section 5.3.1.3)	Provide high bandwidth and improves performance for large data transfers	Cost

Table 5–2: Guidelines for High-Performance Hardware Configurations (cont.)

Hardware	Performance Benefit	Tradeoff
Solid-state disks (Section 5.3.1.4)	Provide very low disk access time	Cost
High-performance host bus adapters (Section 5.3.1.5)	Increase bandwidth and throughput	Cost
DMA host bus adapters (Section 5.3.1.6)	Relieve CPU of data transfer overhead	None
Prestoserve (Section 5.3.1.7)	Improves synchronous write performance	Cost, not supported in a cluster or for nonfile system I/O operations
Hardware RAID subsystem (Section 5.5)	Increases disk capacity and supports write-back cache	Cost of hardware RAID subsystem
Write-back cache (Section 5.3.1.8)	Reduces the latency of many small writes	Cost of hardware RAID subsystem

See the *DIGITAL Systems & Options Catalog* for information about disk, adapter, and controllers performance features.

The following sections describe these guidelines in detail.

5.3.1.1 Using 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 5400 RPM disks can improve performance for many transaction processing applications (TPAs). Extra high-performance 7200 RPM disks are ideal for applications that require both high bandwidth and high throughput.

5.3.1.2 Using Disks with Small Platters

Disks with small platter sizes provide better seek times than disks with large platter sizes, because the disk head has less distance to travel between tracks. There are three sizes for disk platters: 2.5, 3.5, and 5.25 inches in diameter.

A small platter size may improve disk I/O performance (seek time) for applications that perform many small I/O operations, but it provides no performance benefit for large sequential data transfers.

5.3.1.3 Using Disks with Wide Data Paths

Disks with wide (16-bit) data paths provide twice the bandwidth of disks with narrow (8-bit) data paths. Wide disks can improve I/O performance for large data transfers.

5.3.1.4 Using Solid-State Disks

Solid-state disks provide outstanding performance in comparison to regular disks but at a higher cost. Solid-state disks have a disk access time that is less than 100 microseconds, which is equivalent to memory access speed and more than 100 times faster than the disk access time for magnetic disks.

Solid-state disks are ideal for a wide range of response-time critical applications, such as online transaction processing (OLTP), and applications that require high bandwidth, such as video applications. Solid-state disks also provide data reliability through a data-retention system. For the best performance, use solid-state disks for your most frequently accessed data, place the disks on a dedicated bus, and use a high-performance host bus adapter.

5.3.1.5 Using High-Performance Host Bus Adapters

Host bus adapters provide different performance features at various costs. For example, FWD adapters, such as the KZPSA adapter, provide high bandwidth and high throughput connections to disk devices.

SCSI adapters let you set the SCSI bus speed, which is the rate of data transfers. There are three possible bus speeds:

- Slow (up to 5 million bytes per second or 5 MHz)
- Fast (up to 10 million bytes per second or 10 MHz)

  Fast bus speed uses the fast synchronous transfer option, enabling I/O devices to attain high peak-rate transfers in synchronous mode.
- Ultra (up to 20 million bytes per second or 20 MHz)

Not all SCSI bus adapters support all speeds.

5.3.1.6 Using 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.

5.3.1.7 Using Prestoserve

Prestoserve utilizes a nonvolatile, battery-backed memory cache to improve synchronous write performance. Prestoserve temporarily caches 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. You cannot use Prestoserve in a cluster or for nonfile system I/O.

5.3.1.8 Using Write-Back Caches

Hardware RAID subsystems support (as a standard or an optional feature) write-back caches, which can improve I/O write performance while maintaining data integrity. A write-back cache must be battery-backed to protect against data loss and corruption.

A write-back cache decreases the latency of many small writes and can improve write-intensive application performance and Internet server performance. 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.

Because writes appear to be executed immediately, a write-back cache improves performance. If a failure occurs and the cache is battery-backed, upon recovery, the RAID controller will detect any unwritten data that still exists in the write-back cache and write the data to disk before enabling normal controller operations.

5.3.2 Distributing the Disk I/O Load Guidelines

In addition to using hardware that will provide you with the best performance, you must distribute the disk I/O load across devices to obtain the maximum efficiency. Table 5–3 describes guidelines on how to distribute disk I/O and lists the performance benefits as well as tradeoffs.

Table 5–3: Guidelines for Distributing the Disk I/O Load

Action	Performance Benefit	Tradeoff
Distribute swap space across different disks and buses (Section 5.3.2.1)	Improves paging and swapping performance and helps to prevent bottlenecks	Requires additional disks, cabling, and adapters
Distribute disk I/O across different disks and buses (Section 5.3.2.2)	Allows parallel I/O operations and helps to prevent bottlenecks	Requires additional disks, cables, and adapters
Place the most frequently used file systems on different disks (Section 5.3.2.3)	Helps to prevent disk bottlenecks	Requires additional disks
Place data at the beginning of a ZBR disk (Section 5.3.2.4)	Improves bandwidth for sequential data transfers	None

The following sections describe these guidelines in detail.

5.3.2.1 Distributing Swap Space Across Disks and Buses

Distributing swap space across different disks and buses makes paging and swapping more efficient and helps to prevent any single adapter, disk, or bus from becoming a bottleneck. See the *System Administration* manual or swapon(8) for information about configuring swap space.

You can also use LSM to stripe your swap disks, which distributes the disk I/O. See Section 5.4 for more information.

5.3.2.2 Distributing I/O Across Disks and Buses

Distributing disk I/O across different disks and buses helps to prevent a single adapter, disk, or bus from becoming an I/O bottleneck and also allows simultaneous operations.

For example, if you have 16 GB of disk storage, you may get better performance from sixteen 1-GB disks than four 4-GB disks. More spindles (disks) may allow more simultaneous operations. For random I/O operations, 16 disks may be simultaneously seeking instead of 4 disks. For large sequential data transfers, 16 data streams can be simultaneously working instead of 4 data streams.

You can also use LSM to stripe your disks, which distributes the disk I/O load. See Section 5.4 for more information.

5.3.2.3 Distributing File Systems Across Disks

Place the most frequently used file systems on different disks. Distributing file systems will help to prevent a single disk from becoming a bottleneck.

Directories containing executable files or temporary files are often frequently accessed (for example, `/var`, `/usr`, and `/tmp`). If possible, place `/usr` and `/tmp` on different disks.

5.3.2.4 Placing Data at the Beginning of ZBR Disks

Data is most quickly transferred when it is located at the beginning of zone-based recording (ZBR) disks. Placing data at the beginning of these disks improves the bandwidth for sequential data transfers.

5.3.3 General File System Tuning Guidelines

You may be able to improve I/O performance by modifying some kernel attributes that affect overall file system performance. The guidelines apply to all file system configurations, including UFS and AdvFS.

General file system tuning often involves tuning the Virtual File System (VFS). VFS provides a uniform interface that allows common access to files, regardless of the file system on which the files reside.

The file system tuning guidelines fall into these categories:

- Changing how the system allocates and deallocates vnodes

  The kernel data structure for an open file is called a vnode. These are used by all file systems. The allocation and deallocation of vnodes is handled dynamically by the system.

- Increasing the size of the namei cache to make lookup operations faster

  The namei cache is used by all file systems to map file pathnames to inodes.

- Increasing the size of the hash chain table for the namei cache to make lookup operations faster

  Hash tables are used for lookup operations.

- Allocating more memory to the Unified Buffer Cache (UBC)

  The UBC shares physical memory with the virtual memory subsystem and is used to cache the most recently accessed file system data.

- Using Prestoserve to cache only UFS or AdvFS file system metadata

There are also specific guidelines for AdvFS and UFS file systems. See Section 5.6 and Section 5.7 for information.

Table 5–4 describes the guidelines for general file system tuning and lists the performance benefits as well as the tradeoffs.

Table 5–4: Guidelines for General File System Tuning

Action	Performance Benefit	Tradeoff
Increase the maximum number of open files (Section 5.3.3.1)	Allocates more resources to applications	Consumes memory
Increase the size of the namei cache (Section 5.3.3.2)	Improves cache lookup operations	Consumes memory
Increase the size of the hash chain table for the namei cache (Section 5.3.3.3)	Improves cache lookup operations	Consumes memory
Allocate more memory to the UBC (Section 5.3.3.4)	Improves disk I/O performance	May cause excessive paging and swapping
Use Prestoserve to cache only file system metadata (Section 5.3.3.5)	Improves performance for applications that access large amounts of file system metadata	Cost, not supported in a cluster or for nonfile system I/O operations
Cache more vnodes on the free list (Section 5.3.3.6)	Improves cache lookup operations	Consumes memory
Increase the amount of time for which vnodes are kept on the free list (Section 5.3.3.7)	Improves cache lookup operations	None
Delay vnode deallocation (Section 5.3.3.8)	Improves namei cache lookup operations	Consumes memory
Accelerate vnode deallocation (Section 5.3.3.9)	Reduces memory demands	Reduces the efficiency of the namei cache
Disable vnode deallocation (Section 5.3.3.10)	Optimizes processing time	Consumes memory
Increase the open file descriptor limit (Section 5.3.3.11)	Provides more file descriptors to a process	Increases the possibility of runaway allocations
Decrease the open file descriptor limit (Section 5.3.3.11)	Prevents a process from consuming all the file descriptors	May adversely affect the performance of processes that require many file descriptors
Disable clearing of the DMA scatter/gather map registers (Section 5.3.3.12)	Improves performance of VLM/VLDB systems	None

The following sections describe these guidelines in detail.

5.3.3.1 Increasing the Maximum Number of Open Files

Increasing the value of the `max-vnodes` or `maxusers` attribute increases the maximum number of vnodes, which increases the number of open files. If your applications require many open files, you may want to raise the values of these attributes. Raising the attribute values will increase the demand on your memory resources, and should only be done if you get a message stating that you are out of vnodes.

If the number of users on the system exceeds the value of `maxusers`, and you increase the value of `maxusers,` increase the value of `max-vnodes` proportionally.

5.3.3.2 Increasing the Size of the namei Cache

The namei cache is used by all file systems to map file pathnames to inodes. Use `dbx` to monitor the cache by examining the `nchstats` structure. The miss rate (misses / (good + negative + misses)) should be less than 20 percent.

To make lookup operations faster, increase the size of the namei cache by increasing the value of the `maxusers` attribute (the recommended way) or by increasing the value of the `name-cache-size` attribute. Increasing the value of `maxusers` or `name-cache-size` allocates more system resources for use by the kernel. However, it also increases the amount of physical memory consumed by the kernel. Note that many benchmarks may perform better with a large namei cache.

5.3.3.3 Increasing the Size of the Hash Chain Table for the namei Cache

Increasing the size of hash chain table for the namei cache spreads the namei cache elements and may reduce linear searches, which improves lookup speeds. The `name-cache-hash-size` attribute specifies the size of the hash chain table for the namei cache. The default size is 256 slots.

You can change the value of the `name-cache-hash-size` attribute so that each hash chain has three or four name cache entries. To determine an appropriate value for the `name-cache-hash-size` attribute, divide the value of `name-cache-size` attribute by 3 or 4 and then round the result to a power of 2. For example, if the value of `name-cache-size` is 1029, dividing 1029 by 4 produces a value of 257. Based on this calculation, you could specify 256 (2 to the power of 8) for the value of the `name-cache-hash-size` attribute.

5.3.3.4 Allocating More Memory for the UBC

The Unified Buffer Cache (UBC) uses a portion of physical memory to cache actual file system data for reads and writes, AdvFS metadata, and Memory File System (MFS) data. The UBC prevents the system from having to copy data from a disk, which improves performance. If there is an insufficient amount of memory allocated to the UBC, disk I/O performance may be degraded.

Increasing the size of the UBC improves the chance that data will be found in the cache. However, because the UBC and the virtual memory subsystem share the same physical memory pages, increasing the size of the UBC may cause excessive paging and swapping.

See Section 4.8 for information about tuning the UBC.

5.3.3.5 Using Prestoserve to Cache Only File System Metadata

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). If the attribute is set to 1 (enabled), Prestoserve caches only file system metadata.

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.

5.3.3.6 Caching More Free vnodes

You can raise the value of the `min-free-vnodes` attribute, which determines the minimum number of vnodes on the free list. Increasing the value causes the system to cache more free vnodes and improves the performance of cache lookup operations. However, increasing the value will increase the demand on your memory resources.

On 24-MB systems, the default value of the `min-free-vnodes` attribute is 150. On 32-MB or larger systems, the default value depends on the value of the `maxusers` attribute. For these systems, if the value of `min-free-vnodes` is close to the value of the `max-vnodes` attribute, vnode deallocation will not be effective.

If the value of `min-free-vnodes` is larger than the value of `max-vnodes`, vnode deallocations will not occur. If the value of `min-free-vnodes` must be close to the value of `max-vnodes`, you may want to disable vnode deallocation (see Section 5.3.3.10). However, disabling vnode deallocation does not free memory, because memory used by the vnodes is not returned to the system. On systems that need to reclaim the memory used by vnodes, make sure that the value of `min-free-vnodes` is significantly lower than the value of `max-vnodes`.

5.3.3.7 Increasing the Time vnodes Remain on the Free List

You can increase the value of the `vnode-age` attribute to increase the amount of time for which vnodes are kept on the free list. This increases the possibility that the vnode will be successfully looked up. The default value for `vnode-age` is 120 seconds on 32-MB or larger systems and 2 seconds on 24-MB systems.

5.3.3.8 Delaying the Deallocation of vnodes

Increase the value of the `namei-cache-valid-time` attribute to delay the deallocation of vnodes. This can improve namei cache lookup operations but it consumes memory resources.

5.3.3.9 Accelerating the Deallocation of vnodes

Decrease the value of the `namei-cache-valid-time` attribute to accelerate the deallocation of vnodes. This causes vnodes to be deallocated from the namei cache at a faster rate, but reduces the efficiency of the cache.

5.3.3.10 Disabling vnode Deallocation

To optimize processing time, disable vnode deallocation by setting the value of the `vnode-deallocation-enable` attribute to 0. Disabling vnode deallocation does not free memory, because memory used by the vnodes is not returned to the system. You may want to disable vnode allocation if the value of `min-free-vnodes` is close to the value of `max-vnodes`.

5.3.3.11 Modifying the Maximum Number of Open File Descriptors

The `open-max-soft` and `open-max-hard` attributes control the maximum number of open file descriptors for each process. When the `open-max-soft` limit is reached, a warning message is issued, and when the `open-max-hard` limit is reached, the process is stopped. These attributes prevent runaway allocations (for example, allocations within a loop that cannot be exited because of an error condition) from consuming all the available file descriptors.

The `open-max-soft` and `open-max-hard` attributes both have default values of 4096 file descriptors (open files) per process. The maximum number of open files per process is 65,536. If your applications require many open files, you may want to increase the maximum open file descriptor limit. Increasing the limit provides more file descriptors to a process, but it increases the possibility of runaway allocations. In addition, if you increase the number of open files per process, make sure that the `max-vnodes` attribute is set to an adequate value. See the *Release Notes* for information about increasing the open file descriptor limit.

Decreasing the open file descriptor limit decreases the number of file descriptors available to each process and prevents a process from consuming all the file descriptors. However, decreasing the limit may adversely affect the performance of processes that require many file descriptors.

5.3.3.12 Disabling Clearing of the DMA Scatter/Gather Map Registers

If you have an AlphaServer 8200 or 8400, the `dma-sg-map-unload-zero` attribute controls whether the direct memory access (DMA) scatter/gather map registers clear after an I/O operation completes. If your system utilizes large amounts of memory or storage, you may be able to gain some I/O performance benefit by setting the attribute to zero.

5.4 Using the Logical Storage Manager

The Logical Storage Manager (LSM) can improve system performance and provide high data availability. LSM also provides you with online storage management features and enhanced performance information and statistics, with little additional overhead. Although any type of system can benefit from LSM, it is especially suited for large systems with large numbers of disks.

LSM volumes are used in the same way as disk partitions. You can create UFS file systems and AdvFS file domains and filesets on an LSM volume, or you can use a volume as a raw device.

To set up a high-performance LSM configuration, you must be careful how you configure the following:

- Disks, disk groups, and databases (see Section 5.4.1)
- Mirrored disks (see Section 5.4.2)
- Striped disks (see Section 5.4.3)

The *Logical Storage Manager* manual provides detailed information about using LSM. The following sections describe configuration and tuning guidelines for LSM.

5.4.1 Basic LSM Configuration Guidelines

The following sections provide general guidelines to configure LSM disks, disk groups, and databases. How you configure your LSM disks and disk groups determines the flexibility of your LSM configuration.

In addition, each LSM disk group maintains a configuration database, which includes detailed information about mirrored and striped disks and volume, plex, and subdisk records.

Table 5–5 lists LSM disk, disk group, and database configuration guidelines and performance benefits as well as tradeoffs.

Table 5–5: Guidelines for LSM Disks, Disk Groups, and Databases

Action	Benefit	Tradeoff
Initialize your LSM disks as sliced disks (Section 5.4.1.1)	Provides greater storage configuration flexibility	None
Increase the maximum number of LSM volumes (Section 5.4.1.2)	Improves performance on VLM/VLDB systems	None
Make the `rootdg` disk group a sufficient size (Section 5.4.1.3)	Ensure sufficient space for disk group information	None
Use a sufficient private region size for each disk (Section 5.4.1.4)	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 5.4.1.5)	Efficiently utilizes the configuration space	None
Group disks into different disk groups (Section 5.4.1.6)	Allows you to move disk groups between systems	Reduces flexibility when configuring volumes
Use an appropriate size and number of database and log copies (Section 5.4.1.7)	Ensures database availability and improves performance	None
Place disks containing database and log copies on different buses (Section 5.4.1.8)	Improves availability	Cost of additional hardware

The following sections describe these guidelines in detail.

5.4.1.1 Initializing LSM Disks as Sliced Disks

Initialize your LSM disks as sliced disks, instead of as simple disks. A sliced disk provides greater storage configuration flexibility because the entire disk is under LSM control. 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.

5.4.1.2 Increasing the Maximum Number of LSM Volumes

For large systems increase the value of the `max-vol` attribute, which specifies the maximum number of volumes per system. The default is 1024; you can increase it to 4096.

5.4.1.3 Sizing the rootdg Disk Group

You must make sure that the `rootdg` disk group has an adequate size, because the disk group's configuration database contains records for disks outside of the `rootdg` disk group, in addition to the ordinary disk-group configuration information. For example, the `rootdg` configuration database includes disk-access records that define all disks under LSM control.

The `rootdg` disk group must be large enough to contain records for the disks in all the disk groups. See Table 5–6 for more information.

5.4.1.4 Sizing Private Regions

You must make sure that the private region for each disk has an adequate size. LSM keeps disk media label and configuration database copies in each disk's private region.

A private region must be large enough to accommodate the size of the LSM 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. However, a large private region requires more disk space. The default private region size is 1024 blocks, which is usually adequate for configurations using up to 128 disks per disk group.

5.4.1.5 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, in order to efficiently utilize the configuration space. One or two LSM configuration database copies can be stored in a disk's private region.

When you add a new disk to existing an LSM disk group, the size of the private region on the new disk is determined by the private region size of the other disks in the disk group. As you add more disks to a disk group, the `voldiskadd` utility reduces the number of configuration copies and log copies that are initialized for the new disks. See `voldiskadd`(8) for more information.

5.4.1.6 Grouping Disks in Disk Groups

You may want to group disks in disk groups according to their function. This enables disk groups to be moved between systems, and decreases the size of the LSM configuration database for each disk group. However, using multiple disk groups reduces flexibility when configuring volumes.

5.4.1.7 Choosing the Number and Size of the Database and Log Copies

Each disk group maintains a configuration database, which includes detailed information about mirrored and striped disks and volume, plex, and subdisk records. The LSM subsystem's overhead primarily involves managing the kernel change logs and copies of the configuration databases.

LSM performance is affected by the size and the number of copies of the configuration database and the kernel change log. They determine the amount of time it takes for LSM to start up, for changes to the configuration to occur, and for the LSM disks to fail over in a cluster.

Usually, each disk in a disk group contains one or two copies of both the kernel change log and the configuration database. Disk groups consisting of more than eight disks should not have copies on all disks. Always use four to eight copies.

The number of kernel change log copies must be the same as the number of configuration database copies. For the best performance, the number of copies must be the same on each disk that contains copies.

Table 5–6 describes the guidelines for configuration database and kernel change log copies.

Table 5–6: Configuration Database and Kernel Change Log Guidelines

Disks Per Disk Group	Size of Private Region (in Blocks)	Configuration and Kernel Change Log Copies Per Disk
1 to 3	512	Two copies in each private region
4 to 8	512	One copy in each private region
9 to 32	512	One copy on four to eight disks, zero copies on remaining disks
33 to 128	1024	One copy on four to eight disks, zero copies on remaining disks

Table 5–6: Configuration Database and Kernel Change Log Guidelines (cont.)

Disks Per Disk Group	Size of Private Region (in Blocks)	Configuration and Kernel Change Log Copies Per Disk
129 to 256	1536	One copy on four to eight disks, zero copies on remaining disks
257 or more	2048	One copy on four to eight disks, zero copies on remaining disks

5.4.1.8 Distributing the Database and Log Copies Across Buses

For disk groups with large numbers of disks, place the disks that contain configuration database and kernel change log copies on different buses. This provides you with better performance and higher availability.

5.4.2 LSM Mirrored Volume Configuration Guidelines

Use LSM mirrored volumes for high data availability. If a physical disk fails, the mirrored plex (copy) containing the failed disk becomes temporarily unavailable, but the remaining plexes are still available. A mirrored volume has at least two plexes for data redundancy.

Mirroring can also improve read performance. However, a write to a volume results in parallel writes to each plex, so write performance may be degraded. Environments whose disk I/O operations are predominantly reads obtain the best performance results from mirroring. See Table 5–7 for guidelines.

In addition, use block-change logging (BCL) to improve the mirrored volume recovery rate when a system failure occurs by reducing the synchronization time. If BCL is enabled and a write is made to a mirrored plex, BCL identifies the block numbers that have changed and then stores the numbers on a logging subdisk. BCL is not used for reads.

BCL is enabled if two or more plexes in a mirrored volume have a logging subdisk associated with them. Only one logging subdisk can be associated with a plex. BCL can add some overhead to your system and degrade the mirrored volume's write performance. However, the impact is less for systems under a heavy I/O load, because multiple writes to the log are batched into a single write. See Table 5–8 for guidelines.

Note that BCL will be replaced by dirty region logging (DRL) in a future release.

Table 5–7 lists LSM mirrored volume configuration guidelines and performance benefits as well as tradeoffs.

Table 5–7: Guidelines for LSM Mirrored Volumes

Action	Benefit	Tradeoff
Map mirrored plexes across different buses (Section 5.4.2.1)	Improves performance and increases availability	None
Use the appropriate read policy (Section 5.4.2.2)	Efficiently distributes reads	None
Attach up to eight plexes to the same volume (Section 5.4.2.3)	Improves performance for read-intensive workloads and increases availability	Uses disk space inefficiently
Use a symmetrical configuration (Section 5.4.2.4)	Provides more predictable performance	None
Use block-change logging (Table 5–8)	Improves mirrored volume recovery rate	May decrease write performance
Stripe the mirrored volumes (Table 5–9)	Improves disk I/O performance and balances I/O load	Increases management complexity

Table 5–8 lists LSM block-change logging (BCL) configuration guidelines and performance benefits as well as tradeoffs.

Table 5–8: Guidelines for LSM Block-Change Logging

Action	Benefit	Tradeoff
Configure multiple logging subdisks (Section 5.4.2.5)	Improves recovery time	Requires additional disks
Use a write-back cache for logging subdisks (Section 5.4.2.6)	Minimizes BCLs write degradation	Cost of hardware RAID subsystem
Use the appropriate BCL subdisk size (Section 5.4.2.7)	Enables migration to dirty region logging	None
Place logging subdisks on infrequently used disks (Section 5.4.2.8)	Helps to prevent disk bottlenecks	None
Use solid-state disks for logging subdisks (Section 5.4.2.9)	Minimizes BCL’s write degradation	Cost of disks

The following sections describe these guidelines in detail.

5.4.2.1 Mirroring Volumes Across Different Buses

Putting each mirrored plex on a different bus improves performance and availability by helping to prevent bus bottlenecks, and allowing simultaneous I/O operations. Mirroring across different buses also increases availability by protecting against bus and adapter failure.

5.4.2.2 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-robin read

  Satisfies read operations to the volume in a round-robin manner from all plexes in the volume.

- Preferred read

  Satisfies read operations from one specific plex (usually the plex with the highest performance).

- Select

  Selects a default read policy, based on the plex associations to the volume. If the mirrored volume contains a single, enabled, striped plex, the default is to prefer that plex. For any other set of plex associations, the default is to use a round-robin policy.

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, then set the read policy to preferred read for that plex. By default, a mirrored volume with one striped plex should have the striped plex configured as the preferred read. Otherwise, you almost aways use the round-robin read policy.

5.4.2.3 Using Multiple Plexes in a Mirrored Volume

To improve performance for read-intensive workloads, up to eight plexes can be attached to the same mirrored volume. However, this configuration does not use disk space efficiently.

5.4.2.4 Using a Symmetrical Configuration

A symmetrical mirrored disk configuration provides 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.

5.4.2.5 Using Multiple BCL Subdisks

Using multiple block-change logging (BCL) subdisks will improve recovery time after a failure.

5.4.2.6 Using a Write-Back Cache with LSM

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

5.4.2.7 Sizing BCL Subdisks

To support migration from BCL to dirty region logging (DRL), which will be supported in a future release, use the appropriate BCL subdisk size.

If you have less than 64 GB of disk space under LSM control, calculate the subdisk size by multiplying 1 block by each gigabyte of storage. If the result is an odd number, add 1 block; if the result is an even number, add 2 blocks. For example, if you have 1 GB (or less) of space, use a 2-block subdisk. If you have 2 GB (or 3 GB) of space, use a 4-block subdisk.

If you have more than 64 GB of disk space under LSM control, use a 64-block subdisk.

5.4.2.8 Placing BCL Logging Subdisks on Infrequently Used Disks

Place a logging subdisk on an infrequently used disk. Because this subdisk is frequently written, do not put it on a busy disk. Do not configure BCL subdisks on the same disks as the volume data, because this will cause head seeking or thrashing.

5.4.2.9 Using Solid-State Disks for BCL Subdisks

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

5.4.3 LSM Striped Volume Configuration Guidelines

Striping volumes can increase performance because parallel I/O streams can operate concurrently on separate devices. Striping can improve the performance of applications that perform large sequential data transfers or multiple, simultaneous I/O operations.

Striping distributes data across the disks in a volume in stripes with a fixed size. The stripes are interleaved across the striped plex's subdisks, which are located on different disks, to evenly distribute disk I/O.

The performance benefit of striping depends on the stripe width, which is the number of blocks in a stripe, and how your users and applications perform I/O. Bandwidth increases with the number of disks across which a plex is striped. See Table 5–9 for guidelines.

Table 5–9 lists LSM striped volume configuration guidelines and performance benefits as well as tradeoffs.

You may want to combine mirroring with striping to obtain both high availability and high performance. See Table 5–7 and Table 5–9 for guidelines.

Table 5–9: Guidelines for LSM Striped Volumes

Action	Benefit	Tradeoff
Use multiple disks in a striped volume (Section 5.4.3.1)	Improves performance	Decreases volume reliability
Distribute subdisks across different disks and buses (Section 5.4.3.2)	Improves performance and increases availability	None
Use the appropriate stripe width (Section 5.4.3.3)	Improves performance	None
Avoid splitting small data transfers (Section 5.4.3.3)	Improves the performance of volumes that quickly receive multiple data transfers	May use disk space inefficiently
Split large individual data transfers (Section 5.4.3.3)	Improves the performance of volumes that receive large data transfers	Decreases throughput

The following sections discuss these guidelines in detail.

5.4.3.1 Increasing the Number of Disks in a Striped Volume

Increasing the number of disks in a striped volume can increase the bandwidth, depending on the applications and file systems you are using and on the number of simultaneous users. However, this reduces the effective mean-time-between-failures (MTBF) of the volume. If this reduction is a problem, use both striping and mirroring.

5.4.3.2 Distributing Striped Volume Subdisks Across Different Buses

Distribute the subdisks of a striped volume across different buses. This improves performance and helps to prevent a single bus from becoming a bottleneck.

5.4.3.3 Choosing the Correct Stripe Width

The performance benefit of striping depends on the size of the stripe width and the characteristics of the I/O load. Stripes of data are allocated alternately and evenly to the subdisks of a striped plex. A striped plex consists of a number of equal-sized subdisks located on different disks.

The number of blocks in a stripe determines the stripe width. LSM uses a default stripe width of 64 KB (or 128 sectors), which works well in most environments.

Use the `volstat` command to determine the number of data transfer splits. For volumes that receive only small I/O transfers, you may not want to use striping because disk access time is important. Striping is beneficial for large data transfers.

To improve performance of large sequential data transfers, use a stripe width that will divide each individual data transfer and distribute the blocks equally across the disks.

To improve the performance of multiple simultaneous small data transfers, make the stripe width the same size as the data transfer. However, an excessively small stripe width can result in poor system performance.

If you are striping mirrored volumes, ensure that the stripe width is the same for each plex.

5.4.4 LSM Tuning Guidelines

After you set up the LSM configuration, you may be able to improve performance. For example, you can perform the following tasks:

- Balance the I/O load

  LSM allows you to achieve a fine level of granularity in data placement, because LSM provides a way for volumes to be distributed across multiple disks. 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 impacting volume availability.

- Use striping to increase bandwidth for frequently accessed data

  LSM provides a significant improvement in performance when there are multiple I/O streams. After you identify the most frequently accessed file systems and databases, you can realize significant performance benefits by striping the high traffic data across portions of multiple disks, which increases bandwidth to this data.

- Set the preferred read policy to the fastest mirrored plex

  If one plex of a mirrored volume exhibits superior performance, either because the disk is being striped or concatenated across multiple disks, or because it is located on a much faster device, then set the read policy to the preferred read policy for that plex. By default, a mirrored volume with one striped plex should be configured with the striped plex as the preferred read.

- Increase the value of the `volinfo.max_io` parameter. This can improve the performance of systems that use large amounts of memory or storage.

5.5 Hardware RAID Subsystem Configuration Guidelines

Hardware RAID subsystems increase your storage capacity and provide different degrees of performance and availability at various costs. For example, some hardware RAID subsystems support dual-redundant RAID controllers and a nonvolatile write-back cache, which greatly improve performance and availability. Entry-level hardware RAID subsystems provide cost-efficient RAID functionality.

Table 5–10 lists hardware RAID subsystem configuration guidelines and performance benefits as well as tradeoffs.

Table 5–10: Guidelines for Configuring Hardware RAID Subsystems

Hardware	Performance Benefit	Tradeoff
Evenly distribute disks in a storage set across different buses (Section 5.5.1)	Improves performance and helps to prevent bottlenecks	None
Ensure that the first member of each mirrored set is on a different disk	Improves performance	None
Use disks with the same data capacity in each storage set (Section 5.5.2)	Improves performance	None
Use the appropriate chunk size (Section 5.5.3)	Improves performance	None
Stripe mirrored sets (Section 5.5.4)	Increases availability and read performance	May degrade write performance
Use a write-back cache (Section 5.5.5)	Improves write performance	Cost of hardware

Table 5–10: Guidelines for Configuring Hardware RAID Subsystems (cont.)

Hardware	Performance Benefit	Tradeoff
Use dual-redundant RAID controllers (Section 5.5.6)	Improves performance, increases availability, and prevents I/O bus bottlenecks	Cost of hardware
Install spare disks (Section 5.5.7)	Improves availability	Cost of disks
Replace failed disks promptly (Section 5.5.7)	Improves performance	None

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

5.5.1 Distributing Storage Set Disks Across Buses

You can improve performance and help to prevent bottlenecks by distributing storage set disks evenly across different buses.

Make sure that the first member of each mirrored set is on a different bus.

5.5.2 Using Disks with the Same Data Capacity

Use disks with the same capacity in the same storage set.

5.5.3 Choosing the Correct Chunk Size

The performance benefit of stripe sets depends on how your users and applications perform I/O and the chunk (stripe) size. For example, if you choose a stripe size of 8 KB, small data transfers will be distributed evenly across the member disks. However, a 64-KB data transfer will be divided into at least eight data transfers.

You may want to use 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, in order to force the data onto a different disk.

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.

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.

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.

5.5.4 Striping Mirrored Sets

You can stripe mirrored sets to improve performance.

5.5.5 Using a Write-Back Cache

RAID subsystems support, either as a standard or an optional feature, a nonvolatile 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, especially Web servers. 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 because writes appear to be executed 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 HSZ documentation for more information.

5.5.6 Using Dual-Redundant Controllers

If supported, 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.

5.5.7 Using Spare Disks

Install predesignated spare disks on separate controller ports and storage shelves. This will help you to maintain data availability if a disk failure occurs.

5.6 Using the Advanced File System

The Advanced File System (AdvFS) allows you to 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. See the *AdvFS Guide to File System Administration* for more information on using AdvFS.

The following sections describe how to configure and tune AdvFS for high performance.

5.6.1 AdvFS Configuration Guidelines

You will obtain the best performance if you carefully plan your AdvFS configuration. Table 5–11 lists AdvFS configuration guidelines and performance benefits as well as tradeoffs. In addition, the recommendations described in Table 5–2 and Table 5–3 apply to AdvFS configurations.

Table 5–11: AdvFS Configuration Guidelines

Action	Performance Benefit	Tradeoff
Use multiple-volume file domains (Section 5.6.1.1)	Improves throughput and simplifies management	Increases chance of domain failure and may cause log bottleneck
Use several file domains instead of one large domain (Section 5.6.1.1)	Prevents log from becoming a bottleneck	Increases maintenance complexity
Place transaction log on fast or uncongested volume (Section 5.6.1.2)	Prevents log from becoming a bottleneck	None
Preallocate space for the BMT (Section 5.6.1.3)	Prevents prematurely running out of domain space	Reduces available disk space
Increase the number of pages by which the BMT extent size grows (Section 5.6.1.3)	Prevents prematurely running out of domain space	Reduces available disk space
Stripe files (Section 5.6.1.4)	Improves sequential read and write performance	Increases chance of domain failure
Use quotas (Section 5.6.1.5)	Controls file system space utilization	None

The following sections describe these AdvFS configuration guidelines in more detail.

5.6.1.1 Using Multiple-Volume File Domains

Using multiple-volume file domains allows greater control over your physical resources, and may improve a fileset's total throughput. However, be sure that the log does not become a bottleneck. Multiple-volume file domains improve performance because AdvFS generates parallel streams of output using multiple device consolidation queues.

In addition, using only a few file domains instead of using many file domains reduces the overall management effort because fewer file domains require less administration. However, a single volume failure within a file domain renders the entire file domain inaccessible. Therefore, the more volumes that you have in your file domain the greater the risk that a file domain will fail.

DIGITAL recommends that you use a maximum of 12 volumes in each file domain. However, to reduce the risk of file domain failure, limit the number of volumes per file domain to three or use mirrored volumes created with LSM.

For multiple-volume domains, make sure that busy files are not located on the same volume. Use the `migrate` command to move files across volumes.

5.6.1.2 Improving the Transaction Log Performance

Each file domain has a transaction log that keeps track of fileset activity for all filesets in the file domain. The AdvFS file domain transaction log may become a bottleneck. This can occur 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 5.6.2.9 for information on moving an existing transaction log.

To make the transaction log highly available, use LSM to mirror the log.

5.6.1.3 Improving Bitmap Metadata Table Performance

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 that 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.

If a file domain has a large number of small files, you may prematurely run out of disk space for the BMT. Handling many small files makes the system request metadata extents more frequently, which causes the metadata to become fragmented. Because the number of BMT extents is limited, the file domain will appear to be out of disk space if the BMT cannot be extended to map new files.

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-6` 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-12, M-18, 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.

If you are prematurely out of BMT disk space, you may be able to eliminate the problem by defragmenting the file domain that contains the volume. See defragment(8) for more information.

Table 5–12 provides some BMT sizing guidelines for the number of pages to preallocate for the BMT, and the number of pages by which the BMT extent size grows. The BMT sizing depends on the maximum number of files you expect to create on a volume.

Table 5–12: BMT Sizing Guidelines

Estimated Maximum Number of Files on a Volume	Number of Pages to Preallocate	Number of Pages to Grow Extent
< 50,000	3600	128
100,000	7200	256
200,000	14,400	512
300,000	21,600	768
400,000	28,800	1024
800,000	57,600	2048

You can preallocate space for the BMT when the file domain is created, and when a volume is added to the domain by using the mkfdmn and addvol commands with the -p flag.

You can also modify the number of extent pages by which the BMT grows when a file domain is created and when a volume is added to the domain by using the mkfdmn and the addvol commands with the -x flag.

If you use the mkfdmn -x or the addvol -x command when there is a large amount of free space on a disk, as files are created, the BMT will expand by the specified number of pages and those pages will be in one extent. As the disk becomes more fragmented, the BMT will still expand, but the pages will not be contiguous and will require more extents. Eventually, the BMT will run out of its limited number of extents even though the growth size is large.

Using the mkfdmn -p or the addvol -p command to preallocate a large BMT before the disk is fragmented may prevent this problem because the entire preallocated BMT is described in one extent. All subsequent growth will be able to utilize nearly all of the limited number of BMT extents. Do not overallocate BMT space because the disk space cannot be used for other purposes. However, too little BMT space will eventually cause the BMT to grow by a fixed amount. At this time, the disk may be fragmented and the growth will require multiple extents. See `mkfdmn`(8) and `addvol`(8) for more information.

5.6.1.4 Striping Files

The AdvFS `stripe` utility lets you improve the read and write performance of an individual file. This is useful if an application continually accesses a few specific files. See `stripe`(8) for information.

The utility directs a zero-length file (a file with no data written to it yet) to be distributed evenly across several volumes in a file domain. As data is appended to the file, the data is spread across the volumes. AdvFS determines the number of pages per stripe segment and alternates the segments among the disks in a sequential pattern. Bandwidth can be improved by distributing file data across multiple volumes.

Do not stripe both a file and the disk on which it resides.

To determine if you should stripe files, use the `iostat` utility. The blocks per second and I/O operations per second should be cross-checked with the disks bandwidth capacity. If the disk access time is slow, in comparison to the stated capacity, then file striping may improve performance.

5.6.1.5 Using AdvFS Quotas

AdvFS quotas allow you to track and control the amount of physical storage that a user, group, or fileset consumes. AdvFS eliminates the slow reboot activities associated with UFS quotas. In addition, AdvFS quota information is always maintained, but quota enforcement can be activated and deactivated.

For information about UFS quotas, see Section 5.7.1.6.

5.6.2 AdvFS Tuning Guidelines

After you configure AdvFS, you may be able to tune it to improve performance. Table 5–13 lists AdvFS tuning guidelines and performance benefits as well as tradeoffs. In addition, the recommendations described in Table 5–4 apply to AdvFS configurations.

Table 5–13: AdvFS Tuning Guidelines

Action	Performance Benefit	Tradeoff
Increase the percentage of memory allocated for the AdvFS buffer cache (Section 5.6.2.1)	Improves AdvFS performance if data reuse is high	Consumes memory
Defragment file domains (Section 5.6.2.2)	Improves read and write performance	None
Increase the dirty data caching threshold (Section 5.6.2.3)	Improves random write performance	May cause I/O spikes or increase the number of lost buffers if a crash occurs
Decrease the I/O transfer read-ahead size (Section 5.6.2.4)	Improves performance for `mmap` page faulting	None
Disable the flushing of dirty pages mapped with the `mmap` function during a `sync` call (Section 5.6.2.5)	May improve performance for applications that manage their own flushing	None
Modify the AdvFS device queue limit (Section 5.6.2.6)	Influences the time to complete synchronous (blocking) I/O	May cause I/O spikes
Consolidate I/O transfers (Section 5.6.2.7)	Improves AdvFS performance	None
Force all AdvFS file writes to be synchronous (Section 5.6.2.8)	Ensures that data is successfully written to disk	May degrade file system performance
Move the transaction log to a fast or uncongested volume (Section 5.6.2.9)	Prevents log from becoming a bottleneck	None
Balance files across volumes in a file domain (Section 5.6.2.10)	Improves performance and evens the future distribution of files	None
Migrate frequently used or large files to different file domains (Section 5.6.2.11)	Improves I/O performance	None
Decrease the size of the metadata buffer cache to 1 percent (Section 4.7.21)	Improves performance for systems that use only AdvFS	None

The following sections describe how to tune AdvFS in detail.

5.6.2.1 Modifying the Size of the AdvFS Buffer Cache

The `AdvfsCacheMaxPercent` attribute specifies the amount of physical memory that AdvFS uses for its buffer cache.

You may improve AdvFS performance by increasing the percentage of memory allocated to the AdvFS buffer cache. To do this, increase the value of the `AdvfsCacheMaxPercent` attribute. The default is 7 percent of memory, the minimum is 1 percent, and the maximum is 30 percent.

Increasing the value of the `AdvfsCacheMaxPercent` attribute will decrease the amount of memory available to the virtual memory subsystem, so you must make sure that you do not cause excessive paging and swapping. Use the `vmstat` command to check virtual memory statistics.

You may want to increase the AdvFS buffer cache size if data reuse is high. If you increase the value of the `AdvfsCacheMaxPercent` attribute and experience no performance benefit, return to the original value. If data reuse is insignificant or if you have more than 2 GB of memory, you may want to decrease the cache size.

5.6.2.2 Defragmenting a File Domain

AdvFS attempts to store file data in a collection of contiguous blocks (a file extent) on a disk. 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 fragmentation degrades 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 with the `–v` and `–n` options to show the amount of file fragmentation.

The `defragment` utility reduces 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.

You can improve the efficiency of the defragment process by deleting any unneeded files in the file domain before running the `defragment` utility. See `defragment`(8) for more information.

5.6.2.3 Increasing the Dirty Data Caching Threshold for a Volume

Dirty or modified data is data that has been written by an application and cached but has not yet been written to disk. You can increase the amount of dirty data that AdvFS will cache for each volume in a file domain. This can improve write performance for systems that perform many random writes by increasing the number of cache hits.

You can increase the amount of cached dirty data for all new volumes or for a specific, existing volume. The default value is 16 KB. The minimum value is 0, which disables dirty data caching. The maximum value is 32 KB.

If you have high data reuse (data is repeatedly read and written), you may want to increase the dirty data threshold. If you have low data reuse, you may want to decrease the threshold or use the default value.

Use the `chvol -t` command to modify the dirty data threshold for an individual existing volume. You must specify the number of dirty, 512-byte blocks to cache. See `chvol`(8) for more information.

To modify the dirty data threshold for all new volumes, modify the value of the `AdvfsReadyQLim` attribute, which specifies the number of 512-byte blocks that can be on the readylazy queue before the requests are moved to the device queue.

If you change the dirty data threshold and performance does not improve, return to the original value.

5.6.2.4 Decreasing the I/O Transfer Read-Ahead Size

AdvFS reads and writes data by a fixed number of 512-byte blocks. The default is 128 blocks. Use the `chvol` command with the `-w` option to change the write-consolidation size. Use the `chvol` command with the `-r` option to change the read-ahead size. See `chvol`(8) for more information.

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.

If the disk is fragmented so that the pages of a file are not sequentially allocated, reduce fragmentation by using the `defragment` utility. See `defragment`(8) for more information.

5.6.2.5 Disabling the Flushing of Dirty mmapped Pages

A file can have dirty data in memory 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.

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 dirty mmapped pages are asynchronously written to disk. If the `AdvfsSyncMmapPages` attribute is set to 0, dirty mmapped pages are not written to disk during a `sync` system call.

If your applications manage their own `mmap` page flushing, set the value of the `AdvfsSyncMmapPages` attribute to 0.

See `mmap`(2) and `msync`(2) for more information.

5.6.2.6 Modifying the AdvFS Device Queue Limit

Synchronous and asynchronous AdvFS I/O requests are placed on separate consolidation queues, where small, logically contiguous block requests are consolidated into larger I/O requests. The consolidated synchronous and asynchronous I/O requests are moved to the AdvFS device queue and then sent to the device driver.

The `AdvfsMaxDevQLen` attribute limits the AdvFS device queue length. When the number of requests on the device queue exceeds the value of the `AdvfsMaxDevQLen` attribute, only synchronous requests are accepted onto the device queue. The default value of the `AdvfsMaxDevQLen` attribute is 80.

Limiting 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 default value of the `AdvfsMaxDevQLen` attribute is appropriate for most configurations. However, you may need to modify this value if you are using fast or slow adapters, striping, or mirroring. A higher value may improve throughput, but will also increase synchronous read/write time. To calculate response time, multiply the value of the `AdvfsMaxDevQLen` attribute by 9 milliseconds (the average I/O latency).

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.

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).

5.6.2.7 Consolidating I/O Transfers

Consolidating a number of I/O transfers into a single, large I/O transfer can improve AdvFS performance. To do this, use the `chvol` command with the `–c on` flag. This is the default. DIGITAL recommends that you do not disable the consolidation of I/O transfers. See `chvol`(8) for more information.

5.6.2.8 Forcing Synchronous Writes

Use the `chfile -l on` command to force all write requests to an AdvFS file to be synchronous. When forced synchronous write requests to a file are enabled, the `write` system call returns a success value only after the data has been successfully written to disk. This may degrade file system performance.

When forced synchronous write requests to a file are disabled, the `write` system call returns a success value when the requests are cached. The data is then written to disk at a later time (asynchronously).

5.6.2.9 Moving the Transaction Log

Make sure that the AdvFS transaction log resides on an uncongested disk and bus or system performance may be degraded.

If the transaction log becomes a bottleneck, 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. 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. See `switchlog`(8) and `showfdmn`(8) for more information.

In addition, you can divide the file domain into several smaller file domains. This will cause each domain's transaction log to handle transactions for fewer filesets.

5.6.2.10 Balancing a Multivolume File Domain

If the files in a multivolume domain are not evenly distributed, performance may be degraded. The `balance` utility distributes the percentage of used space evenly between volumes in a multivolume file domain. This improves performance and evens 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. See balance(8) for more information.

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). See showfdmn(8) for more information.

5.6.2.11 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. However, using the balance utility after migrating files may cause the files to move to a different volume. See balance(8) for more information.

In addition, a file that is migrated is defragmented at the same time, if possible. Defragmentation makes the file more contiguous, which improves performance. Therefore, you can use the migrate command to defragment selected files. See migrate(8) for more information.

5.7 Using the UNIX File System

The following sections will help you to configure and tune UNIX File Systems (UFS).

5.7.1 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 tune some parameters by using the tunefs command.

Table 5–14 describes UFS configuration guidelines and performance benefits as well as tradeoffs. In addition, the recommendations described in Table 5–2 and Table 5–3 apply to UFS configurations.

Table 5–14: UFS Configuration Guidelines

Action	Performance Benefit	Tradeoff
Increase the file system fragment size to 8 KB (Section 5.7.1.1)	Improves performance for large files	Wastes disk space for small files
Use the default file system fragment size of 1 KB (Section 5.7.1.1)	Uses disk space efficiently	None
Reduce the density of inodes (Section 5.7.1.2)	Improves performance of large files	None
Allocate blocks contiguously (Section 5.7.1.3)	Aids UFS block clustering	None
Increase the number of blocks combined for a read (Section 5.7.1.4)	Improves performance	None
Use a Memory File System (MFS) (Section 5.7.1.5)	Improves I/O performance	Does not ensure data integrity because of cache volatility
Use disk quotas (Section 5.7.1.6)	Controls disk space utilization	UFS quotas may slow reboot time

The following sections describe the UFS configuration guidelines in detail.

5.7.1.1 Modifying the File System Fragment Size

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 block size, which is 8 KB. Use the `newfs` command to do this.

However, to use disk space efficiently, use the default fragment size, which is 1 KB. See `newfs`(8) for more information.

5.7.1.2 Reducing the Density of inodes

The number of files in a file system is determined by the number of inodes and the size of the file system. The default is to create an inode for each 4096 bytes of data space.

If a file system will contain many large files, you may want to increase the amount of data space allocated to an inode and reduce the density of inodes. To do this, use the `newfs -i` command to specify the amount of data space allocated to an inode. See `newfs`(8) for more information.

5.7.1.3 Allocating Blocks Contiguously

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 and aid UFS block clustering. You can do this by using either the `tunefs` command or the `newfs` command. See `newfs`(8) and `tunefs`(8) for more information.

5.7.1.4 Increasing the Number of Blocks Combined for a Read

The value of the UFS `maxcontig` parameter specifies the number of blocks that can be combined into a single cluster. The default value of `maxcontig` is 8 KB. The file system attempts read operations in a size that is defined 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.

Use the `tunefs` command or the `newfs` command to change the value of `maxcontig`. See `newfs`(8) and `tunefs`(8) for more information.

5.7.1.5 Using a Memory File System

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 file system can improve read/write performance, but it is a volatile cache. The contents of an MFS file system are lost after a reboot, unmount operation, or power failure.

Because no date is written to disk, an MFS file system is a very fast file system and can be used to store temporary files or read-only files that are loaded into it 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 file system to cache the temporary files that are created during the build and reduce the build time.

5.7.1.6 Using UFS Disk Quotas

You can specify UFS file system limits for user accounts and for groups by setting up file system quotas, also known as disk 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 slow reboot time. For information about AdvFS quotas, see Section 5.6.1.5.

5.7.2 UFS Tuning Guidelines

After you configure your UFS file systems, you can modify some parameters and attributes to improve performance. Table 5–15 describes UFS tuning guidelines and performance benefits as well as tradeoffs. In addition, the recommendations described in Table 5–4 apply to UFS configurations.

Table 5–15: UFS Tuning Guidelines

Action	Performance Benefit	Tradeoff
Increase size of metadata buffer cache to more than 3 percent of main memory (Section 4.9.1)	Increases cache hit rate and improves UFS performance	Requires additional memory resources
Defragment the file system (Section 5.7.2.1)	Improves read and write performance	Requires down time
Delay flushing full write buffers to disk (Section 5.7.2.2)	Frees CPU cycles	May degrade real-time workload performance
Increase number of blocks combined for read ahead (Section 5.7.2.3)	Improves performance	None
Increase number of blocks combined for a write (Section 5.7.2.4)	Improves performance	None
Increase the maximum number of UFS or MFS mounts (Section 5.7.2.5)	Allows more mounted file systems	None

The following sections describe how to tune UFS in detail.

5.7.2.1 Defragmenting a File System

When a file consists of many discontiguous 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.

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 `dbx` to examine the `ufs_clusterstats`, `ufs_clusterstats_read`, and `ufs_clusterstats_write` structures. See `dbx`(1) for more information.

UFS block clustering is usually efficient. If the numbers from the UFS clustering kernel structures show that clustering is not being particularly effective, the files in the file system may be very fragmented.

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.

AdvFS provides you with the ability to defragment a file domain by using the `defragment` command. See `defragment`(8) for more information.

5.7.2.2 Delaying Full Write Buffer Flushing

You can free CPU cycles by using the `dbx` debugger to set the value of the `delay_wbuffers` kernel variable to 1, which delays flushing full write buffers to disk at the next `sync` call. However, this may adversely affect real-time workload performance. The default value of `delay_wbuffers` is 0. See `dbx`(1) for more information.

5.7.2.3 Increasing the Number of Blocks Combined for Read Ahead

You can increase the number of blocks that are combined for a read-ahead operation.

To do this, use the `dbx` debugger to make the value of the `cluster_consec_init` kernel variable equal to the value of the `cluster_max_read_ahead` variable (the default is 8), which specifies the maximum number of read-ahead clusters that the kernel can schedule. See `dbx`(1) for more information.

In addition, you must make sure that cluster read operations are enabled on nonread-ahead and read-ahead blocks. To do this, the value of the `cluster_read_all` kernel variable must be set to 1 (the default).

5.7.2.4 Increasing the Number of Blocks Combined for a Write

The `cluster_maxcontig` parameter specifies the number of blocks that are combined into a single write operation. The default value is 8 KB. Contiguous writes are done in a unit size that is determined by the file system block size (the default is 8 KB) multiplied by the value of the `cluster_maxcontig` parameter.

5.7.2.5 Increasing the Number of UFS or MFS Mounts

Mount structures are dynamically allocated when a mount request is made and subsequently deallocated when an unmount request is made. The `max-ufs-mounts` attribute specifies the maximum number of UFS and MFS mounts on the system.

You can increase the value of the `max-ufs-mounts` attribute if your system will have more than the default limit of 1000 mounts.

5.8 Tuning CAM

DIGITAL UNIX uses the Common Access Method (CAM) as 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 can modify the following attributes:

- `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.

6 Tuning the Network Subsystem

This chapter describes the guidelines to tune networks and the Network File System (NFS). Many of the tuning tasks described in this chapter require you to modify system attributes. See Section 2.11 for more information about attributes.

6.1 Tuning Networks

Most resources used by the network subsystem are allocated and adjusted dynamically; however, there are some tuning recommendations that you can use to improve performance, particularly with systems that are Internet 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 congestion to occur:

- A problem with one or more components of the network (hardware or software)
- A workload (network traffic) that consistently exceeds the capacity of the available resources even though everything is 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.

To obtain the best network performance, you must understand your workload and the performance characteristics of your network hardware, as described in Chapter 1 and the *DIGITAL Systems & Options Catalog*. Different network interfaces have different performance characteristics, including raw performance and system overhead. For example, a Fiber Distributed Data Interface (FDDI) interface provides better performance than an Ethernet interface.

Before you can tune your network, you must determine whether the source of the performance problem is an application, network interconnect, network controller, or the communication partner. Table 6–1 lists network subsystem tuning guidelines and performance benefits as well as tradeoffs.

Table 6–1: Network Tuning Guidelines

Action	Performance Benefit	Tradeoff
Increase the size of the hash table that the kernel uses to look up TCP control blocks (Section 6.1.1)	Improves the TCP control block lookup rate and increases the raw connection rate	Slightly increases the amount of wired memory
Increase the limits for partial TCP connections on the socket listen queue (Section 6.1.2)	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 maximum number of concurrent nonreserved, dynamically allocated ports (Section 6.1.3)	Allows more simultaneous outgoing connections	Negligible increase in memory usage
Enable TCP keepalive functionality (Section 6.1.4)	Enables inactive socket connections to time out	None
Increase the size of the kernel interface alias table (Section 6.1.5)	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 6.1.6)	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 6.1.7)	Frees connection resources sooner	Reducing the timeout limit increases the potential for data corruption, so guideline should be applied with caution
Reduce the TCP retransmission rate (Section 6.1.8)	Prevents premature retransmissions and decreases congestion	A long retransmit time is not appropriate for all configurations
Enable the immediate acknowledgement of TCP data (Section 6.1.9)	Can improve network performance for some connections	May adversely affect network bandwidth

Table 6–1: Network Tuning Guidelines (cont.)

Action	Performance Benefit	Tradeoff
Increase the TCP maximum segment size (Section 6.1.10)	Allows sending more data per packet	May result in fragmentation at router boundary
Increase the size of the transmit and receive buffers for a TCP socket (Section 6.1.11)	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 6.1.12)	Helps to prevent dropping UDP packets	May decrease available memory when the buffer space is being used
Allocate sufficient memory to the UBC (Section 6.1.13)	Improves disk I/O performance	May decrease the physical memory available to the virtual memory subsystem
Disable the use of a PMTU (Section 6.1.14)	Improves the efficiency of Web servers that handle remote traffic from many clients	May reduce server efficiency for LAN traffic

The following sections describe these tuning guidelines in detail.

6.1.1 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 `tcbhashsize` attribute specifies the number of hash buckets in the kernel TCP connection table (the number of buckets in the `inpcb` hash table). 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 and improve performance.

The default value is 32. For Web servers and proxy servers, set the `tcbhashsize` attribute to 16384.

6.1.2 Tuning the Socket Listen Queue Limits

You may be able to improve performance by increasing the limits for the socket listen queue (only for TCP). The `somaxconn` attribute 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 default value of the somaxconn attribute is 1024; the maximum value is 65535.

To improve throughput and response time with fewer drops, you can increase the value of the somaxconn attribute. A busy system running applications that generate a large number of connections (for example, a Web server) may have many pending connections. For these systems, set the value of the somaxconn attribute to the maximum value of 65535.

The sominconn attribute 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 default value is 0. The value of the sominconn attribute overrides the application-specific backlog value, which may be set too low for some server software. To improve performance without recompiling an application, you can set the value of the sominconn attribute to the maximum value of 65535. The value of the sominconn attribute should be the same as the value of the somaxconn attribute.

Network performance can degrade if a client saturates a socket listen queue with erroneous TCP SYN packets, effectively blocking other users from the queue. To eliminate this problem, 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 tcp_keepinit attribute to 30 (15 seconds).

Three socket subsystem attributes monitor socket listen queue events:

- The sobacklog_hiwat attribute counts the maximum number of pending requests to any server socket.
- The sobacklog_drops attribute counts the number of backlog drops that exceed the socket set backlog.
- The somaxconn_drops attribute counts the number of drops that exceed the value of the somaxconn attribute.

Use the sysconfig -q socket command to display the kernel variable values. If the values show that the queues are overflowing, you may need to increase the socket listen queue limit. See Section 2.9.3 for information about monitoring the sobacklog_hiwat, sobacklog_drops, and somaxconn_drops attributes.

6.1.3 Increasing the Maximum Number of Concurrent Nonreserved Dynamically Allocated Ports

The ipport_userreserved attribute controls the number of times you can simultaneously make outgoing connections to other systems. The number of outgoing ports is the value of the `ipport_userreserved` attribute minus 1024. The default value of the attribute is 5000; therefore, the default number of outgoing ports is 3976. The maximum value of the `ipport_userreserved` attribute is 65535.

When the kernel dynamically allocates a nonreserved port number for use by a TCP or UDP application that creates an outgoing connection, it selects the port number from a range of values between 1024 and the value of the `ipport_userreserved` attribute. Because each TCP client must use one of these ports, the range limits the number of simultaneous outgoing connections to a value specified by the value of the attribute minus 1024.

If your system requires many outgoing ports, you may need to increase the value of the `ipport_userreserved` attribute. If your system is a proxy server with a load of more than 4000 connections, increase the value of the `ipport_userreserved` attribute to 65535.

DIGITAL does not recommend reducing the value of the `ipport_userreserved` attribute to a value that is less than 5000.

6.1.4 Enabling TCP keepalive Functionality

Keepalive functionality enables the periodic transmission of messages on a connected socket in order to keep connections active. If you enable keepalive, 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, set the `tcp_keepalive_default` attribute 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 `tcp_keepidle` attribute specifies the amount of idle time before keepalive probes in 0.5 second units. The default interval is 2 hours.
- The `tcp_keepintvl` attribute specifies the amount of time between retransmission of keepalive probes in 0.5 second units. The default interval is 75 seconds.
- The `tcp_keepcnt` attribute specifies the maximum number of keepalive probes that are sent before the connection is dropped. The default is 8 probes.

- The `tcp_keepinit` attribute specifies the maximum amount of time before an initial connection attempt times out in 0.5 second units. The default is 75 seconds.

6.1.5 Improving the Lookup Rate for IP Addresses

The `inifaddr_hsize` attribute specifies the number of hash buckets in the kernel interface alias table (`in_ifaddr`). The default value of the `inifaddr_hsize` attribute is 32; the maximum value is 512.

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. Increasing the number of hash buckets in the table can improve performance on systems that use large numbers of aliases.

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.

6.1.6 Decreasing the Partial TCP Connection Timeout Limit

The `tcp_keepinit` attribute is the amount of time that a partially established TCP connection remains on the socket listen queue before it times out. The value of the attribute is in units of 0.5 seconds. The default value is 150 units (75 seconds).

Partial connections consume listen queue slots and fill the queue with connections in the `SYN_RCVD` state. You can make partial connections time out sooner by decreasing the value of the `tcp_keepinit` attribute. However, do not set the value 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). If you have a 32000 socket queue limit, the default (75 seconds) is usually adequate.

Network performance can degrade if a client overfills a socket listen queue with TCP SYN packets, effectively blocking other users from the queue. To eliminate this problem, increase the value of the `sominconn` attribute to the maximum of 64000. If the system continues to drop SYN packets, decrease the value of the `tcp_keepinit` attribute to 30 (15 seconds).

6.1.7 Decreasing the TCP Connection Context Timeout Limit

You can make the TCP connection context time out more quickly at the end of a connection. However, this will increase the chance of data corruption.

The 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 `tcp_msl` attribute determines the maximum lifetime of a TCP segment and the timeout value for the `TIME_WAIT` state.

The value of the 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. DIGITAL strongly recommends using the default value; otherwise, there is the potential for data corruption.

6.1.8 Decreasing the TCP Retransmission Rate

The `tcp_rexmit_interval_min` attribute specifies the minimum amount of time between the first TCP retransmission. For some wide area networks (WANs), the default value may be too small, causing premature retransmission timeouts. This may lead to duplicate transmission of packets and the erroneous invocation of the TCP congestion-control algorithms.

The `tcp_rexmit_interval_min` attribute is specified in units of 0.5 seconds. The default value is 1 unit (0.5 seconds).

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. However, not every connection needs a long retransmission time. Usually, the default value is adequate. Do not specify a value that is less than 1 unit. Do not change the attribute unless you fully understand TCP algorithms.

6.1.9 Disabling Delaying the Acknowledgment of TCP Data

The value of the `tcpnodelack` attribute determines whether the system delays acknowledging TCP data. The default is 0, which delays the acknowledgment of TCP data. Usually, the default is adequate. However, for some connections (for example, loopback), the delay can degrade performance. You may be able to improve network performance by setting the value of the `tcpnodelack` attribute to 1, which disables the acknowledgment delay. However, this may adversely impact network bandwidth. Use the `tcpdump` command to check for excessive delays.

6.1.10 Increasing the Maximum TCP Segment Size

The `tcp_mssdflt` attribute specifies the TCP maximum segment size (the default value of 536). You can increase the value to 1460. This allows sending more data per socket, but may cause fragmentation at the router boundary.

6.1.11 Increasing the Transmit and Receive Buffers for a TCP Socket

The `tcp_sendspace` attribute specifies the default transmit buffer size for a TCP socket. The `tcp_recvspace` attribute specifies the default receive buffer size for a TCP socket. The default value of both attributes is 32 KB. You can increase the value of these attributes to 60 KB. This 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).

6.1.12 Increasing the Transmit and Receive Buffers for a UDP Socket

The `udp_sendspace` attribute specifies the default transmit buffer size for an Internet User Datagram Protocol (UDP) socket; the default value is 9 KB. The `udp_recvspace` attribute specifies the default receive buffer size for a UDP socket; the default value is 40 KB. You can increase the values of these attributes to 64 KB. However, increasing the values uses more memory when the buffers are being used by an application (sending or receiving data).

6.1.13 Allocating Sufficient Memory to the UBC

You must ensure that you have sufficient memory allocated to the Unified Buffer Cache (UBC). Servers that perform lots of file I/O (for example, Web and proxy servers) extensively utilize both the UBC and the virtual memory subsystem. In most cases, use the default value of 100 percent for the `ubc-maxpercent` attribute, which specifies the maximum amount of physical memory that can be allocated to the UBC. If necessary, you can decrease the size of the attribute by increments of 10 percent.

See Section 4.8 for more information about tuning the UBC.

6.1.14 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 `pmtu_enabled` attribute is enabled (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.

On a Web server that handles local traffic and some remote traffic, enabling the use of a PMTU can improve bandwidth. However, if a Web server handles traffic among many remote clients, enabling the use of a PMTU can cause an excessive increase in the size of the kernel routing tables, which can reduce server efficiency. If a Web server has poor performance and the routing table increases to more than 1000 entries, set the value of the `pmtu_enabled` attribute to 0 to disable the use of PMTU protocol.

6.2 Tuning the Network File System

The Network File System (NFS) shares the unified buffer cache with the virtual memory subsystem and local file systems. Most performance problems with NFS can be attributed to bottlenecks in the virtual memory, network, or disk subsystem.

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` command. 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.

Table 6–2 lists NFS tuning guidelines and performance benefits as well as tradeoffs.

Table 6–2: Guidelines for NFS Tuning

Action	Performance Benefit	Tradeoff
Use Prestoserve (Section 6.2.1)	Improves synchronous write performance	Cost
Use the appropriate number of `nfsd` daemons on the server (Section 6.2.2)	Enables efficient I/O blocking operations	None

Table 6–2: Guidelines for NFS Tuning (cont.)

Action	Performance Benefit	Tradeoff
Use the appropriate number of `nfsiod` daemons on the client (Section 6.2.3)	Enables efficient I/O blocking operations	None
Increase the number of I/O threads (Section 6.2.4)	May improve NFS read and write performance	None
Modifying cache timeout limits (Section 6.2.5)	May improve network performance for read-only file systems and slow network links	None
Decrease network timeouts (Section 6.2.6)	May improve performance for slow or congested networks	None
Use NFS protocol Version 3.0 (Section 6.2.7)	Improves network performance	Decreases the performance benefit of Prestoserve

The following sections describe these guidelines in detail.

6.2.1 Using Prestoserve to Improve 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 stable (nonvolatile) 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.

6.2.2 Using the Appropriate Number of nfsd Daemons

Servers use `nfsd` daemons to handle NFS requests from client machines. The number of `nfsd` daemons determines the number of parallel operations and must be a multiple of 8. For good performance on frequently used NFS servers, configure a network with either 16 or 32 `nfsd` daemons. Having exactly 16 or 32 `nfsd` daemons produces the most efficient blocking for I/O operations.

6.2.3 Using the Appropriate Number of nfsiod Daemons

Clients use `nfsiod` daemons to service asynchronous I/O operations such as buffer cache readahead and delayed write operations. The number of `nfsiod` daemons determines the number of outstanding I/O operations. The number of `nfsiod` daemons 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 `nfsiod` daemons (plus 1) is the number of writes that a client can have outstanding at any one time. Having exactly 7 or 15 `nfsiod` daemons produces the most efficient blocking for I/O operations. If write gathering is enabled, and the client is not running any `nfsiod` daemons, you may experience a performance degradation. To disable write gathering, use `dbx` to set the `nfs_write_gather` kernel variable to 0.

6.2.4 Increasing the Number of Threads

On a client system, the `nfsiod` daemons spawn several I/O threads to service asynchronous I/O operations 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.

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 Chapter 2, `nfsiod`(8), and `nfsd`(8) for more information.

6.2.5 Modifying Cache Timeout Limits

For read-only file systems and slow network links, performance may be improved by changing the cache timeout limits. 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 give you slightly better 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.

6.2.6 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 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 total calls, NFS performance may be severely degraded. See Chapter 2 for sample `nfsstat` output of timeout and call statistics and `nfsstat`(8) for more information.

You can also use the `netstat -s` command to verify the existence of a timeout problem. A nonzero count for `fragments dropped after timeout` in the `ip` section of the `netstat` output may indicate that the problem exists. See Chapter 2 for sample `netstat` command output.

If fragment drops are a problem, 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.

6.2.7 Using NFS Protocol Version 3.0

NFS protocol Version 3.0 provides client-side asynchronous write support, which improves client perception of performance, improves the cache consistency protocol, and requires less network load than Version 2. Protocol Version 3 decreases the performance benefit of Prestoserve.

A
Tuning Special Configurations

This appendix provides information about tuning special configurations. See Section 2.11 for information about modifying system attributes.

A.1 Tuning Internet Servers

Internet servers require that you modify the default values of some system attributes. Internet servers include World Wide Web servers, proxy servers, mail servers, and ftp servers. See Chapter 6 for detailed information about these attributes.

You can modify the following `socket` subsystem attributes and specify the values as indicated:

- `somaxconn` = 65535
- `sominconn` = 65535

You can modify the following `inet` subsystem attributes and specify the values as indicated:

- `tcphashsize` = 16384
- `ipport_userreserved` = 65535
- `pmtu_enabled`

You can modify the following `vm` subsystem attributes and specify the values as indicated:

- `ubc-maxpercent` = 100
- `vm-mapentries` = 20000
- `vm-maxvas` = 10737418240
- `vm-vpagemax` = 131072

You can modify the following `proc` subsystem attributes and specify the values as indicated:

- `maxusers` = 512
- `max-proc-per-user` = 512
- `max-threads-per-user` = 4096

A.2 Tuning a Low-Memory Workstation

The following sections describe tuning considerations for low-memory (24-MB) Alpha systems. Some of these tuning considerations may also apply to DIGITAL UNIX workstations in general, regardless of size.

A.2.1 Attribute Settings for Low-Memory Workstations

The following attribute settings are automatically used when installing DIGITAL UNIX on a 24-MB Alpha system:

```
generic:
 lite-system=1

proc:
 ncallout_alloc_size=4096

vfs:
 bufcache=2
 max-vnodes=1000
 min-free-vnodes=150
 vnode-age=2
 namei-cache-valid-time=30
 name-cache-size=150

io:
 bdevsw-size = 70
 cdevsw-size = 125
 max-iosize-read = 65536
 max-iosize-write = 65536
 basic-dma-window-size = 0
 cam_ccb_pool_size = 100
 cam_ccb_low_water = 50
 cam_ccb_increment = 25

network:
 arptab_nb=19

vm:
 vm-aggressive-swap = 1
```

The default attribute settings for 24-MB Alpha systems increase the amount of physical memory available to user applications by reducing the amount of memory used for system caches. These settings may also work well on 32-MB or larger Alpha systems that are being used as personal workstations (that is, not being used as timesharing systems or file servers). You can apply the settings to your workstation by entering the following command:

```
# sysconfigdb -f /etc/sysconfigtab.lite -m
```

The settings can be removed by entering the following command:

```
# sysconfigdb -f /etc/sysconfigtab.lite -r
```

After entering either of these `sysconfigdb` commands, you must reboot the system to apply the new attribute values.

A.2.2 Swap Space and Memory Tuning on Low-Memory Systems

Swapping can cause problems with low-memory systems. When operating in **deferred** mode (also referred to as overcommittment or lazy mode), low-memory systems will use more swap space. Low-memory systems have to overcommit more physical memory than high-memory systems. As a result, low-memory systems will do more pageouts and will use more swap space.

When swap space is exhausted, you will receive warning messages, and processes will be killed. The only solution to this problem is to increase either memory or swap space. See Chapter 4 for information on swap modes. (Low-memory systems operating in **immediate** mode do not have special problems with swapping.)

Low-memory systems can also have special problems with the Unified Buffer Cache (UBC). If `vmstat` output shows excessive pageins but few or no pageouts, the value of the `ubc-borrowpercent` attribute may be too small. It is particularly important to watch for this on low-memory systems, because they tend to reclaim UBC pages more aggressively than systems with more memory, and this condition can have an adverse effect on system performance. See Chapter 4 for information about attribute settings affecting the UBC.

A.2.3 X Window System Considerations for Low-Memory Workstations

On low-memory systems, you may want to consider the following adjustments that affect memory use by the X Window system:

- If a shortage of memory (and the paging and swapping associated with limited memory) is slowing the performance of your system, you may need to reset the X server to free up memory resources.

  Space that has been allocated to the X server is never freed to the system until the X server is terminated; it can be reused by the X server, but it is not freed. As a result, the amount of memory resources allocated to the X server is the largest amount of memory resources ever used by the X server at any single point in time in either the current session or any prior session. This amount can be rather large if at some point you opened a large number of windows, displayed PostScript text, and performed other windowing operations that consume a lot of memory.

To reset the X server and eliminate any reserve of unused memory that may have built up over time, you need to specify the –`terminate` option in the `Xserver.conf` file (`/var/X11/Xserver.conf`) and then kill the X server (`kill` *`pid`*). The X Display Manager, `xdm`, will automatically restart your X server. (Use the `ps` command to locate the process identification (PID) number for the X server.) With the –`terminate` option in effect, the X server will now reset upon each logout.

The `Xserver.conf` file contains information on how to add options. For example, the entry for the –`terminate` option, with no other options present in the entry, is as follows:

```
args < -terminate >
```

If you are running `xdm`, you can establish the –`terminate` option in either the `Xserver.conf` file or the `Xservers` file (`/var/X11/xdm/Xservers`).

- On low-memory systems, compile Motif applications with the –`shared` option (the default) instead of the –`non_shared` option. Applications compiled with the –`shared` option have smaller executable files and consume fewer system resources than applications compiled with the –`non_shared` option. This is especially true for Motif applications; for example, a Motif application that is 400 KB with the –`shared` option may be as large as 4 MB with the –`non_shared` option.

B
Configuration Attribute Definitions

This appendix lists all the attributes in the `sysconfigtab` file that can be modified by using the Common Desktop Environment Kernel Tuner (`dxkerneltuner`), the `sysconfig` command, and the `sysconfigdb` command. See Section 2.11 for information on modifying attributes.

The attributes in this appendix are grouped by subsystem. Before you can modify an attribute, you must know the subsystem to which the attribute belongs. This appendix describes attributes for the following subsystems:

- `advfs`—Advanced File System (AdvFS) subsystem (Section B.1)
- `bsd_tty`—TTY subsystem (Section B.2)
- `cm`—Configuration Manager subsystem (Section B.3)
- `dli`—Data Link Interface (DLI) subsystem (Section B.4)
- `generic`—Generic kernel subsystem (Section B.5)
- `inet`—Internet subsystem (Section B.6)
- `io`—I/O subsystem (Section B.7)
- `ipc`—Interprocess Communication (IPC) subsystem (Section B.8)
- `lsm`—Logical Storage Manager (LSM) subsystem (Section B.9)
- `net`—Network subsystem (Section B.10)
- `presto`—Prestoserve subsystem (Section B.11)
- `proc`—Process subsystem (Section B.12)
- `pts`—Pseudoterminal subsystem (Section B.13)
- `rt`—Real-time subsystem (Section B.14)
- `sec`—Security subsystem (Section B.15)
- `snmpinfo`—Simple network management protocol (SNMP) information subsystem (Section B.16)
- `socket`—Socket subsystem (Section B.17)
- `streams`—STREAMS subsystem (Section B.18)
- `ufs`—UNIX File System (UFS) subsystem (Section B.19)
- `vfs`—Virtual File System (VFS) subsystem (Section B.20)

- vm—Virtual memory subsystem (Section B.21)
- xpr—XPR subsystem (Section B.22)

This is not a complete list of the subsystems that have attributes in the sysconfigtab file. Subsystems not included in this appendix include the following: bufcall, cam, cam_disk, cam_tape, cma_dd, ddr, eisa, kds, kinfo, kio, ldtty, ppp, qvision, strstd, table_mgr, timod, tirdwr, vga, ws, and xtiso.

Some attributes have corresponding parameters whose values are specified in the system configuration file. In this appendix, if an attribute has a corresponding parameter, the name of the parameter appears in parentheses after the attribute name. If possible, always modify the attribute instead of its corresponding parameter.

In addition, attributes that can be modified at run time by using the Kernel Tuner or the sysconfig −r command are preceded by an asterisk (*****).

B.1 AdvFS Subsystem Attributes

The AdvFS (advfs) subsystem attributes are as follows:

AdvfsCacheMaxPercent

Determines the percentage of system memory that is allocated to the AdvFS buffer cache. The minimum value is 1 percent. The maximum value is 30 percent.

Default value: 7 percent.

AdvfsMaxFragGrps

Fragment group deallocation starts when the number of fragment groups on the free list is higher than this value. The minimum value is 4. The maximum value is 8192.

Default value: 48

AdvfsMinFragGrps

Fragment group deallocation stops when the number of fragment groups on the free list is less than this value. The minimum value is 3. The maximum value is 8191.

Default value: 16

AdvfsSyncMmapPages

This 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 dirty mmapped pages are asynchronously written to disk. If the `AdvfsSyncMmapPages` attribute is set to 0, dirty mmapped pages are not written to disk during a `sync` system call.

Default value: 1

`AdvfsMaxDevQLen`
This attribute limits the length of the AdvFS device queue. The AdvFS device queue accepts both asynchronous I/O requests and synchronous I/O requests. When the number of requests on the device queue exceeds the value of the `AdvfsMaxDevQLen` attribute, only synchronous requests are accepted onto the device queue.

The minimum value is 0. The maximum value is 65536.

The default value should be appropriate for most configurations. However, you may need to modify this value for systems with very fast or very slow devices and adapters. One 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. If you do not want to limit the size of the device queue, set the value of the `AdvfsMaxDevQLen` parameter to 0 (zero).

Default value: 80

`AdvfsReadyQLim`
This attribute specifies the number of 512-byte blocks that can be on the readylazy queue before the requests are moved to the device queue. The minimum value is 0, which disables buffering on the readylazy queue. The maximum value is 32 KB.

Default value: 16 KB

`AdvfsAccessMaxPercent`
Specifies the maximum percentage of the `malloc` pool that can be allocated for access structures. The minimum value is 5. The maximum value is 95.

Default value: 80

`AdvfsAccessCleanupPercent`
When the percentage of access structures on the closed list reaches this value, structures are reclaimed from the closed list to populate the free list. The minimum value is 5. The maximum value is 95.

Default value: 33

`AdvfsFavorBlockingQueue`

Controls the movement of I/O requests from the consolidation queue to the device queue. When enabled (the default), AdvFS first issues synchronous I/O. When disabled (set to 0), asynchronous I/O is flushed to disk, regardless of synchronous I/O. DIGITAL recommends that you use the default value.

Default value: 1 (enabled)

B.2 TTY Subsystem Attribute

The TTY (`bsd_tty`) subsystem attribute is as follows:

`nclist` (`nclist`)

Controls the number of clist buffers allocated.

The TTY subsystem is specifically for the clist-based TTY subsystem. (DIGITAL UNIX has both STREAMS-based and clist-based TTY subsystems.) The clist-based TTY subsystem is used by the console and the serial ports.

Note that is is also possible to configure the pseudoterminal subsystem to be clist-based. This is accomplished by specifying the pseudodevice `pty` *nn* (default: *nn* = 80) in the kernel configuration file.

B.3 Configuration Manager Subsystem Attribute

The Configuration Manager (`cm`) subsystem attribute is as follows:

`max_callbacks`

Specifies the maximum number of registered callbacks that are allowed by the kernel at any point in time. If exceeded, an error message is issued.

B.4 DLI Subsystem Attributes

The Data Link Interface (DLI) (`dli`) subsystem attributes are as follows:

`must-be-root`

When enabled, requires the user of the raw socket interface to be root. Disabling `must-be-root` allows any user to use the raw socket interface to DLI.

DECnet and STREAMS protocol stacks use a DLI-provided raw interface to the data link. DLI also provides a raw socket interface.

Default value: 1 (enabled)

`max-ifq-length`
: Specifies the number of received packets that are queued to DLI. If many packets are dropped, you may want to increase the value.

  Default value: 512

B.5 Generic Kernel Subsystem Attributes

The generic kernel (`generic`) subsystem attributes are as follows:

`binlog-buffer-size`
: Overrides the kernel buffer size. If the `binlog-buffer-size` attribute is not specified, the kernel buffer size is assigned a value at boot time, based on the amount of physical memory installed in the system. The assigned size ranges from 32 KB to 1 MB. Using the `binlog-buffer-size` attribute, you can specify a minimum size of 8 KB, and a maximum size of 1 MB.

`booted_args`
: The arguments passed from `osf_boot` to the kernel.

`booted_kernel`
: The name of the kernel that was loaded by `osf_boot`.

`clock-frequency`
: The rate of clock interrupts per second.

  Default value: 1024

`cpu-enable-mask` (`cpu_enable_mask`)
: A bit mask that determines whether secondary CPUs can be started by the master CPU. The lowest order bit (bit 0) in this mask corresponds to CPU 0, which is usually the master CPU. The next highest order bit corresponds to CPU 1, and so on.

  The default value for this mask is -1, which means that all CPUs present in the system are allowed to be started.

  You can specify 0 for this mask to enable uniprocessor operation in a multi-CPU system. (The bit for the master CPU does not need to be set because the master CPU is always started. This bit is always set automatically at boot time.)

  If `lockmode` has been specified as 0 or 1, all bits in the `cpu_enable_mask` bit mask that do not correspond to the master CPU are set to 0.

  Default value: -1

`dump-sp-threshold`

Creates multiple-partition dumps or allows dumps to be placed on the primary swap partition, if possible. If a dump will fit on the primary swap partition and leave space that is equal to the threshold value, the dump is created as a single-volume dump on the primary swap partition, even if secondary swap partitions are available. (See the *Kernel Debugging* manual for details.)

Default value: 4096

`kmem_debug`

An interactive boot flag that is used for diagnostic purposes only.

If enabled (1), each time the kernel memory allocator allocates or deallocates memory in the kernel memory pool, the system checks whether the operation is performed correctly. If the kernel memory pool is in a corrupt state, the system crashes and provides useful debugging information.

`lite-system`

When enabled (1), applies values to various attributes in order to improve performance for 24-MB systems.

`lockdebug` (`lockdebug`)

Controls lock debugging. If lock debugging is enabled (`lockdebug=1`), the default value for `lockmode` will be 4. However, if you specify a value for the `lockmode` attribute, this value will override the `lockdebug` value. For example, if the value of `lockmode` is not 4 (that is, 0 to 3), the value of `lockdebug` will be 0. If the value of `lockmode` is 4, the value of `lockdebug` will be 1.

Default value: 0 (disabled)

`lockmode` (`lockmode`)

The mode of the simple lock primitive package within the kernel. Locking primitives support the following combinations of real-time (RT) kernel preemption, symmetric multiprocessing (SMP), and lock debugging with lock statistics:

- When neither RT nor SMP is required (`lockmode=0`), the calls to the simple lock primitives are patched out completely.
- When only RT is required (`lockmode=1`), the simple lock operations maintain a “preemption blocking” count.
- When SMP is required (`lockmode=2`), the lock operations provide synchronization between multiple CPUs.

- When both RT and SMP are required (`lockmode=3`), both sets of processing (`lockmode=1` and `lockmode=2`) are performed.
- When lock debugging and stats are required (`lockmode=4`), all of the processing for `lockmode=3` is performed, with the addition of kernel lock debugging and statistics.

The default value for `lockmode` is assigned at boot time depending on the values for `rt_preempt_opt`, `cpu_enable_mask`, and `lockdebug`, and on whether multiple CPUs are able to be booted. If a value for `lockmode` is specified (0-4), it overrides the default setting and disables any of the three related capabilities that cannot be supported by the chosen set of simple lock primitives.

`lockmaxcycles`
Used internally for debugging purposes. Do not modify.

`locktimeout` (`locktimeout`)
The number of seconds that a CPU will wait (spin) on a simple lock. If a CPU cannot acquire a simple lock in the specified amount of time, a fatal error occurs, and the system panics and issues a “simple_lock: time limit exceeded” message.

Default value: 15 (seconds)

`max-lock-per-thread`
The depth to which complex locks can be nested for a thread at one time. The value of `max-lock-per-thread` is used for debugging (in `lockmode=4`).

`message-buffer-size` `msgbuf_size`
The size of the message buffer that is used to store boot log messages.

Default value: 4096

`old_obreak`
Internal use only. Do not modify.

Default value: 1 (on)

`physio_max_coalescing`
Do not modify.

Controls how I/O requests are gathered when `readb` or `writeb` operations are directed at a character device (not a block device). For example, when set to 65536, eight 8-KB buffers coalesce into one 64-KB buffer. This improves the efficiency of database operations.

`rt-preempt-opt` (`rt_preempt_opt`)
: Controls whether real-time kernel preemption is enabled. If real-time kernel preemption is enabled (1), the default value for `lockmode` will be 1 on a single-CPU system or 3 on a multi-CPU system. However, if a value for `lockmode` has been specified as 0 or 2, this value overrides the `rt_preempt_opt` value and assign `rt_preempt_opt` the value of 0.

  Default value: 0 (disabled)

`user_cfg_pt`
: Internal use only. Do not modify.

  Default value: 0 (disabled)

B.6 Internet Subsystem Attributes

The Internet (`inet`) subsystem attributes are as follows:

`inifaddr_hsize`
: Specifies the number of hash buckets in the kernel interface alias table (`in_ifaddr`). The value of the `inifaddr_hsize` attribute is always rounded down to the nearest power of 2. The maximum value is 512.

  Default value: 32

`ipdefttl`
: The default Internet Protocol (IP) time-to-live value.

  Default value: `DEFTTL`

`ipdirected_broadcast`
: Enables (1) or disables (0) a check to determine whether an IP datagram whose destination address is a directed broadcast address has been received on the interface corresponding to that broadcast address.

  Default value: 0 (disabled)

`ipforwarding`
: When enabled (1), causes a system to forward IP packets that are not addressed to the system. This functionality is usually enabled by using the `/usr/sbin/iprsetup` command.

  Default value: 0 (disabled)

`ipfragttl`
: Maximum time an IP fragment can spend waiting to be reassembled.

  Default value: `IPFRAGTTL` (in units of .05 seconds)

`ipgateway`
: When enabled (1), causes a system to forward IP packets that are not addressed to the system. This functionality is usually enabled by using the `/usr/sbin/iprsetup` command.

  Default value: 0 (disabled)

`ipport_userreserved`
: Specifies the number of times you can simultaneously make outgoing connections to other systems. The number of outgoing ports is the value of the `ipport_userreserved` attribute minus 1024. The default value of the attribute is 5000; therefore, the default number of outgoing ports is 3976. The maximum value of the `ipport_userreserved` attribute is 65535.

  Default value: 5000

`ipsendredirects`
: Enables (1) or disables (0) sending ICMP redirect messages.

  Default value: 1 (enabled)

`ipsrcroute`
: Enables (1) or disables (0) source routing.

  Default value: 1 (enabled)

`pmtu_enabled`
: Enables (1) or disables (0) path maximum transfer unit (PMTU) discovery.

  Default value: 1 (enabled)

`pmtu_decrease_intvl`
: Time to wait after a decrease in a PMTU value before attempting to determine if the PMTU value has increased.

  Default value: `PMTU_DECREASE_INTVL` (in units of .05 seconds)

`pmtu_increase_intvl`
: Time to wait after an increase in a PMTU value before attempting to determine if the PMTU has increased.

  Default value: `PMTU_INCREASE_INTVL` (in units of .05 seconds) `pmtu_rt_check_intvl`

Timer processing interval for routes participating in the PMTU discovery process.

Default value: `PMTU_RT_CHECK_INTVL` (in units of .05 seconds)

`subnetsarelocal`

When enabled (1), considers local all IP addresses that are in the same network but in a different subnet. When disabled (0), considers local only IP addresses that match a directly connected subnet.

Default value: 1 (enabled)

`tcbhashsize`

Number of buckets in the Transmission Control Protocol (TCP) `inpcb` hash table.

Default value: 32

`tcp_compat_42`

Enables (1) or disables (0) 4.2 BSD-compatible behavior for the initial send sequence of numbers and keepalives.

Default value: 1 (enabled)

`tcp_dont_winscale`

Enables (0) or disables (1) window scaling.

Default value: 0 (enabled)

`tcp_keepalive_default`

When set to 1, enables TCP keepalive for all sockets. Use the `tcp_keepalive_default` attribute to override programs that do not set keepalive on their own or you do not have access to the application sources.

Keepalive enables the periodic transmission of messages on a connected socket in order to keep connections active. If keepalive is enabled, 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.

Default value: 0 (disabled)

`tcp_keepcnt`

Maximum number of keepalive probes that can be sent before a connection is dropped.

Default value: `TCPTV_KEEPCNT`

`tcp_keepidle`
Idle time before the first keepalive probe.

Default value: `TCPTV_KEEP_IDLE` (in units of .05 seconds)

`tcp_keepinit`
Initial connect timeout.

Default value: `TCPTV_KEEP_INIT` (in units of .05 seconds)

`tcp_keepintvl`
Time between keepalive probes.

Default value: `TCPTV_KEEP_INTVL` (in units of .05 seconds)

`tcp_mssdflt`
Default maximum segment size.

Default value: `TCP_MSS` (bytes)

`tcpnodelack`
Enables (0) or disables (1) delayed acknowledgments.

Default value: 0 (enabled)

`tcp_msl`
Determines the maximum lifetime of a TCP segment.

Default value: 60 units (30 seconds)

`tcp_recvspace`
Default receive buffer size for TCP sockets.

Default value: `TCP_RECVSPACE` (bytes)

`tcp_rexmit_interval_min`
Minimum amount of time between TCP retransmissions.

Default value: 1 unit (0.5 seconds)

`tcprexmtthresh`
Number of duplicate acknowledgments (ACKs) before retransmission.

Default value: `TCPREXMTTHRESH`

`tcp_rttdflt`
Initial assumed round-trip time.

Default value: 3 (seconds)

`tcp_sendspace`
: Default send buffer size for TCP sockets.

  Default value: `TCP_SENDSPACE` (bytes)

`tcp_ttl`
: IP time-to-live for TCP packets.

  Default value: `TCP_TTL`

`tcptwreorder`
: Enables (1) or disables (0) the movement of TCP `inpcbs` in the `TIME_WAIT` state to the end of the `inpcb` list.

  Default value: 1 (enabled)

`tcp_urgent_42`
: Enables (1) or disables (0) 4.2 BSD-compatible behavior for an urgent pointer. When enabled, the urgent pointer is a pointer to the first octet of data past the urgent section. When disabled, the urgent pointer is a pointer to the last octet of data in the urgent section.

  Default value: 1 (enabled)

`udpcksum`
: Enables (1) or disables (0) Internet user datagram protocol (UDP) checksumming.

  Default value: 1 (enabled)

`udp_recvspace`
: Default receive buffer size for UDP sockets.

  Default value: `UDP_RECVSPACE` (bytes)

`udp_sendspace`
: Default send buffer size for UDP sockets.

  Default value: `UDP_SENDSPACE` (bytes)

`udp_ttl`
: IP time-to-live for UDP packets.

  Default value: `UDP_TTL`

B.7 I/O Subsystem Attributes

The I/O (`io`) subsystem attributes are as follows:

`basic-dma-window-size`

Used to control the allocation of Direct Memory Access (DMA) bus space for AlphaServer 2100 systems that use high-performance cluster interconnects. To ensure adequate bus space for this hardware configuration, set the attribute to 256. For other systems or for AlphaServer 2100s without cluster interconnects, adequate bus space is provided by the usual allocation scheme, which allocates DMA bus space as a percentage of the physical memory on the system.

Default value: 0 (MB)

`bdevsw_size` (`nblkdev`)

Size of the in-memory table for the `bdev` switch. This table is used to access device drivers, and must be large enough to accommodate the drivers registered in the `bdevsw` table in the `/usr/sys/`*`system_name`*`/conf.c` file and any additional device drivers that will be dynamically loaded.

Default value: 70

`cdevsw_size` (`nchrdev`)

Size of the in-memory table for the `cdev` switch. This table is used to access device drivers and must be large enough to accommodate the drivers registered in the `cdevsw` table in the `/usr/sys/`*`system_name`*`/conf.c` file and any additional device drivers that will be dynamically loaded.

Default value: 125

`cam_ccb_pool_size` (`cam_ccb_pool_size`)

Initial size of the buffer pool free list at boot time. One Common Access Method (CAM) control block (ccb) is needed for each outstanding I/O request. Do not modify.

Default value = 200

`cam_ccb_low_water` (`cam_ccb_low_water`)

More buffers are allocated from the kernel when the number of buffers in the pool free list falls below this value. CAM reserves the number of buffers specified by the `cam_ccb_low_water` attribute to ensure that the kernel always has enough memory to shut down runaway processes. Do not modify.

Default value: 100

`cam_ccb_increment` (`cam_ccb_increment`)

Number of buffers either added to or removed from the buffer pool free list. Buffers are allocated as they are needed in order to handle immediate demands; however, they are deallocated carefully to prevent spikes. Do not modify.

Default value: 50 buffers

`device_switch_inited`

Internal use only. Do not modify.

`device_switch_print`

Internal use only. Do not modify.

`device_switch_stale`

Internal use only. Do not modify.

`max-iosize-read`

When nondriver kernel code needs to know the maximum size of an I/O request for a block-I/O device, it issues an `ioctl` call to the driver. If that `ioctl` fails or if the value returned is 0, the value of the `max-iosize-read` attribute is used.

Change the default value of this attribute only if a third-party device driver does not use the `ioctl` call and has a maximum size that is less than the default value in the `sysconfigtab` file (64 KB). (The person writing the third-party device driver must provide information, either in documentation or an installation script, on how to change the entry in the `sysconfigtab` file to the correct value.)

Default value: 65536

`max-iosize-write`

When nondriver kernel code needs to know the maximum size of an I/O request for a block-I/O device, it issues an `ioctl` call to the driver. If that `ioctl` fails or if the value returned is 0, the value of the `max-iosize-write` attribute is used.

Change the default value of this attribute only if a third-party device driver does not use the `ioctl` call and has a maximum size that is less than the default value in the `sysconfigtab` file (64 KB). (The person writing the third-party device driver must provide information, either in documentation or an installation script, on how to change the entry in the `sysconfigtab` file to the correct value.)

Default value: 65536

B.8 IPC Subsystem Attributes

The Interprocess Communication (IPC) (`ipc`) subsystem attributes are as follows:

`max-kernel-ports` (`port_max_num`)
: The maximum number of kernel IPC ports that can be used on the system at one time.

  Default value: (`task-max` * 3 + `thread-max`) + (`thread-max` * 2) + 2000

  (Values of variables used to establish default value: `task-max` = `nproc` +1; `thread-max` = `nproc` *2; `nproc` = 20 + 8 * `maxusers`)

`msg-max` (`msgmax`)
: Maximum size of a single System V message.

  Default value: 8192 bytes (1 page)

`msg-mnb` (`msgmnb`)
: Maximum number of bytes that can be queued to a single System V message queue.

  Default value: 16384 bytes (2 pages)

`msg-mni` (`msgmni`)
: Maximum number of System V message queues that can be used on the system at one time.

  Default value: 50 (The system rounds the number to the value associated with the next higher power of two; for example, 64.)

`msg-tql` (`msgtql`)
: Maximum number of messages that can be queued to a single System V message queue at one time.

  Default value: 40 messages

`num-of-sems` (`num_of_sems`)
: Obsolete. Not used.

`port-hash-max-num` (`port_hash_max_num`)
: Number of port hash buckets that the kernel uses to manage the kernel ports.

  Do not modify.

  Default value: 50 * `max-kernel-ports`

`port-reserved-max-num` (`port_reserved_max_num`)
: Maximum number of ports that can be reserved by the kernel.

  Do not modify.

  Default value: `max-kernel-ports`

`sem-aem` (`semaem`)
: Maximum adjustment that can be made to any System V semaphore when a process exits.

  Default value: 16384

`sem-mni` (`semmni`)
: Maximum number of System V semaphores that can be used on the system at one time.

  Default value: 10 (The system rounds the number to the value associated with the next higher power of two; for example, 16.)

`sem-msl` (`semmsl`)
: Maximum number of System V semaphores that can be used by a single process at one time.

  Default value: 25

`sem-opm` (`semopm`)
: Maximum number of operations that can be outstanding on a single System V semaphore at one time.

  Default value: 10

`sem-ume` (`semume`)
: Maximum number of undo operations that can be outstanding on a single System V semaphore at one time.

  Default value: 10

`sem-vmx` (`semvmx`)
: Maximum integer value that any System V semaphore can contain.

  Default value: 32767

`set-max-num` (`set_max_num`)
: Maximum number of port sets that can be used by the kernel at one time. Do not modify.

  Default value: `task-max` + `thread-max` + 200

  (Values of variables used to establish default value: `task-max` = `nproc` +1; `thread-max` = `nproc` *2; `nproc` = 20 + 8 * `maxusers`)

`shm-max` (`shmmax`)
: Maximum size, in bytes, of a single System V shared memory region.

  Default value: 4194304 bytes (512 pages)

`shm-min` (`shmmin`)
: Minimum size, in bytes, of a single System V shared memory region.

  Default value: 1 (All requests are rounded to the next page size.)

`shm-mni` (`shmmni`)
: Maximum number of shared memory regions that can be used on the system at one time.

  Default value: 100 (The system rounds the number to the value associated with the next higher power of two; for example, 128.)

`shm-seg` (`shmseg`)
: Maximum number of System V shared memory regions that can be attached to a single process at one time.

  Default value: 32

***** `ssm-enable-core-dump` (`ssm_enable_core_dump`)
: If enabled (1), writes segmented shared memory contents when an application issues a core dump. Because segmented shared memory can be large, the amount of time needed to dump the region to a core file and the amount of file system space required by the operation can be extensive, especially in large database environments. Therefore, although shared memory can be useful for debugging, you may not want to include it in core files because of time and resource limitations.

  This attribute can be modified at run time.

  Default value: 1 (enabled)

***** `ssm-threshold` (`ssm_threshold`)
: When not 0, specifies the minimum size of a System V shared region for the use of shared page tables. Setting this value to 0 disables the use of shared page tables for shared memory. The size must be at least equal to the value of `SSM_SIZE`, which is defined in the `machine/pmap.h` file (the default is 8 MB).

  This attribute can be modified at run time.

  Default value: `SSM_SIZE`

B.9 LSM Subsystem Attributes

The Logical Storage Manager (LSM) (`lsm`) subsystem attributes are as follows:

`lsm_rootdev_is_volume`
: Used by the subsystem when the root file system is on an LSM volume. Do not modify.

`lsm_swapdev_is_volume`
: Used by the subsystem when the primary swap device is on an LSM volume. Do not modify.

`max-vol`
: Maximum number of LSM volumes per system. The maximum number of volumes is 4096.

  Default value: 1024

B.10 Network Subsystem Attributes

The network (`net`) subsystem attributes are as follows:

`arptab_nb`
: Number of hash buckets in the address resolution protocol (ARP) table. For optimal hashing, the number should be a prime number.

  Default value: 37

`netisrthreads`
: Number of network threads configured in a system.

  Default value: *n* (Based on the number of CPUs in a system. For a system with one processor, the value is 1. For a multiprocessing system, the value is 1 plus the number of processors.)

`nslip`
: Number of serial line internet protocol (SLIP) lines.

  Default value: 1

B.11 Prestoserve Subsystem Attribute

The Prestoserve (`presto`) attribute is as follows:

`prmetaonly` (`prmetaonly`)
: Controls whether Prestoserve will cache only UFS and AdvFS file-system metadata. If the attribute is set to 1 (enabled), Prestoserve caches only file-system metadata instead of both metadata and synchronous write data. This capability 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.

  The Prestoserve product consists of optional hardware (NVRAM) and software that must be installed on your system. See the *Guide to Prestoserve* for details on Prestoserve.

  Default value: 0 (disabled)

B.12 Process Subsystem Attributes

The process (`proc`) subsystem attributes are as follows:

`autonice` (`autonice`)
: When enabled (1), applications that use more than 600 seconds of CPU time will automatically increase their `nice` values (that is, lower their scheduling priorities).

  Default value: 0 (disabled)

`autonice-penalty`
: The `nice` value that is assigned to a process after it has used an amount of CPU time that exceeds the value of the `autonice-time` attribute.

  Default value: 4

`autonice-time`
: The amount of CPU time, in seconds, that a process can use before it is assigned the `nice` value that is specified by the `autonice-penalty` attribute.

  Default value: 600

`enhanced-core-max-versions`
: Specifies the maximum number of unique core files that a program can create on a host system. The miminum value is 1. The maximum value is 99,999.

  Default value: 16

`enhanced-core-name`

When enabled (1), this attribute allows you to create multiple core file versions. When disabled, core files will be overwritten.

Default value: 0 (disabled)

`give-boost` (`give_boost`)

When enabled (1), this attribute boots the priority of processes that have recently awakened from a block I/O operation. This reduces I/O latency and may make the system more responsive.

Default value: 1 (enabled)

`max-per-proc-address-space` (`vm_initial_limit_vas.rlim_max`)

Maximum amount of user process address space.

Default value: 107374182 (1 GB)

`max-per-proc-data-size` (`vm_initial_limit_data.rlim_max`)

Maximum size of a data segment for each process.

Default value: 107374182 (1 GB)

`max-per-proc-stack-size` (`vm_initial_limit_stack.rlim_max`)

Specifies the maximum size of a user process stack.

Default value: 33554432

`max-proc-per-user` (`maxuprc`)

Maximum number of processes (tasks) that a user can create. (The superuser is not affected.)

Default value: 64

`max-threads-per-user` (`maxuthreads`)

Maximum limit of threads a user can create. (The superuser is not affected.)

`maxusers` (`maxusers`)

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

Increasing the value of the `maxusers` attribute allocates more system resources to the kernel. However, it also increases the amount of physical memory consumed by the kernel. Changing the value of the `maxusers` attribute affects the values of other attributes, including the `taskmax`, `threadmax`, and `min-free-vnodes` attributes.

Default value: System dependent

`ncallout` (`ncallout`)
Obsolete. Not used.

`ncallout_alloc_size`
Minimum amount of memory that can be used for timeout tables. The value of this attribute is automatically adjusted. Do not modify.

`open-max-hard` (`open_max_hard`)
Hard limit for the number of file descriptors for each process. If the number of file descriptors reaches the value of the `open-max-hard` attribute or higher, the process is stopped. Use the `getdtablesize` system call to obtain the total number of file descriptors in a process' descriptor table.

Default value: 4096

`open-max-soft` (`open_max_soft`)
Specifies the soft limit for the number of file descriptors for a process. When the `open-max-soft` limit is reached, a warning message is issued. Use the `getdtablesize` system call to obtain the total number of file descriptors in a process' descriptor table. A process can increase its soft limit up to its hard limit (`open-max-hard`) by using the `setrlimit` system call.

Default value: 4096

`per-proc-address-space` (`vm_initial_limit_vas.rlim_cur`)
Specifies the maximum amount of user process address space.

Default value: 107374182 (1 GB)

`per-proc-data-size` (`vm_initial_limit_data.rlim_cur`)
Current maximum size of a data segment for each process.

Default value: 134217728

`per-proc-stack-size` (`vm_initial_limit_stack.rlim_cur`)
Specifies the maximum size of a user process stack.

Default value: 2097152

`round-robin-switch-rate` (`round_robin_switch_rate`)
Number of context switches per second that can occur between processes with the same priority. The lower the number, the less the

system timeslices; the higher the number, the more the system timeslices.

`sched-min-idle` (`sched_min_idle`)
: Time that a thread must remain idle on a multiprocessor system before it is eligible to migrate to another processor. This attribute is used to tune the soft affinity algorithm on multiprocessor systems. This enables a process to stay where it last ran, and thereby optimize its use of any data or instructions that it had brought into cache memory.

  The `sched-min-idle` attribute is used for multiprocessor systems; it has no effect on single-CPU systems.

`task-max` (`taskmax`)
: Maximum number of tasks that can run simultaneously on the system.

  Default value: 20 + 8 * `maxusers`

`thread-max` (`threadmax`)
: Maximum number of kernel threads that can run simultaneously on the system.

  Default value: 2 * `task-max`

B.13 Pseudoterminal Subsystem Attribute

The pseudoterminal (`pts`) subsystem attribute is as follows:

`nptys`
: Number of STREAMS-based pseudoterminals (`ptys`) that can be active on a system.

  Default value: 255

B.14 Real-Time Subsystem Attributes

The real-time (`rt`) subsystem attributes are as follows:

`aio-max-num`
: Maximum number of concurrent asynchronous I/O (AIO) requests that can be outstanding on the system at one time.

  Default value: 716

`aio-max-percent`

Percentage of physical memory that the asynchronous I/O (AIO) database can occupy. This limits the maximum number of concurrent asynchronous I/O requests that can be set by the `aio-max-num` attribute.

Default value: 1 (percent)

`aio-percpu-data`

When disabled (0), the system's asynchronous I/O (AIO) resources are consolidated into a single database, allowing a single process to launch a number of simultaneous AIO requests up to the value of the `aio-max-num` attribute.

When enabled (1), the AIO database is distributed across the CPUs in the system. This improves access to AIO resources for systems running multiple processes that make simultaneous AIO requests.

The appropriate setting for the `aio-percpu-data` attribute depends on the characteristics of your database application.

Default value: 0 (disabled)

`aio-max-retry`

Number of times that the subsystem, when doing raw I/O to a device, will retry an attempt to lock a user I/O buffer into memory before returning a failure (`EAGAIN`).

Default value: 0

`aio-retry-scale`

Controls progression of wait periods between asynchronous I/O retries.

Default value: 4

`aio-task-max-num`

Maximum number of simultaneous asynchronous I/O requests that can be outstanding per task.

Default value: 333

`sigqueue-max-num`

Maximum number of signal-queuing operations that can be outstanding per process.

Default value: 64

B.15 Security Subsystem Attributes

The security (`sec`) subsystem attributes are as follows:

`audit-buffer-size`

Size of the audit buffer in 1-KB units. If the audit overhead is heavy, I/O performance may decline. Increasing the audit buffer size reduces I/O activity by a small amount. However, if you route audit data to a slow device (for example, to a tape device) or across a network, increasing the audit buffer size can impact I/O activity.

The minimum size is 16 KB; the maximum size is 1024 KB.

Default value: 16 KB

`audit-site-events`

The size, in bytes, reserved for the audit site mask. Each byte can support four site events.

Default value: 16 (bytes)

`nfs-flatten-mode`

Controls the permission bits of a file with access control lists (ACLs) as seen by an NFS Version 2 client. You can specify the following values for the `nfs-flatten-mode` attribute:

- 0—unmodified setting. The actual file mode is sent.
- 1—restrictive setting. The group and other fields of the file mode are modified so that only access that would be granted to everyone in the ACL is granted by the mode bits.
- 2—permissive setting. The group and other bits are modified so that access that would be granted to anyone in the ACL is granted by the mode bits.

See `proplistd`(8) for more information.

Default value: 0

B.16 SNMP Information Subsystem Attribute

The Simple Network Management Protocol (SNMP) information (`snmpinfo`) subsystem attribute is as follows:

`snmp_devs`

Number of times SNMP agents can concurrently access the `snmpinfo` mechanism to obtain kernel information.

Default value: 12

B.17 Socket Subsystem

The socket (`socket`) subsystem attributes are as follows:

`sominconn`
: Minimum length of the socket listen queue (backlog). The value of the `sominconn` attribute overrides the application-specific backlog if the value is greater than the backlog. The `sominconn` attribute has priority over the `somaxconn` attribute.

  The maximum value of the `sominconn` attribute is 65535.

  Default value: 0

`somaxconn`
: Maximum length of the socket listen queue (backlog). The value of the `somaxconn` attribute overrides the application-specific backlog if the value is less than the backlog. The `sominconn` attribute has priority over the `somaxconn` attribute.

  The maximum value of the `somaxconn` attribute is 65535.

  Default value: 1024

`sb_max`
: Maximum size of the socket buffer.

  Default value: `SB_MAX`

`sobacklog_hiwat`
: A read-only attribute that shows the maximum number of pending requests to any of the server sockets in the system. The initial value is 0.

`sobacklog_drops`
: A read-only attribute that is incremented when a received SYN packet is dropped because the number of queued SYN_RCVD connections for a socket is equal to that socket's backlog limit. The initial value is 0.

`somaxconn_drops`
: A read-only attribute that is incremented when a received SYN packet is dropped because the number of queued SYN_RCVD connections for a socket is equal to the value of the `somaxconn` attribute. The initial value is 0.

B.18 STREAMS Subsystem Attributes

The STREAMS (`streams`) subsystem attributes are as follows:

`nstrpush`

Maximum number of STREAMS modules that can be pushed on a stream. This prevents an errant user process from consuming all the available queues on a single stream.

Default value: 9

`strmsgsz`

Maximum size, in bytes, of the data portion of a STREAMS message. This prevents a single `write` or `putmsg` from consuming too many message blocks.

Default value: 12288 (12 KB)

B.19 UFS Subsystem Attributes

The UNIX File System (`ufs`) subsystem attributes are as follows:

`create-fastlinks` (`create_fastlinks`)

Enables (1) or disables (0) the creation of fast symbolic link files.

Default value: 1 (enabled)

`inode-hash-size` (`inohsz`)

Size of the inode hash chain table for the inode least recently used (LRU) cache.

Large inode hash chain tables spread the inode structures and may make chain lengths short. This can reduce linear searches and improve lookup speeds. In general, chains should contain only 2 or 3 elements.

The maximum value is `max-vnodes`/2.

Default value: 512 (slots)

`nmount` (`nmount`)

Obsolete. Replaced by the `vfs` subsystem's `max-ufs-mounts` attribute.

`ufs-blkpref-lookbehind` (`ufs_blkpref_lookbehind`)

Range of blocks behind the current block location through which to search for a free block to allocate for an indirect block write operation (for all writes other than the first).

A value greater than 1 will enable a look-behind search before writing each indirect block after the first write.

Default value: 8

B.20 VFS Subsystem Attributes

The Virtual File System (vfs) subsystem attributes are as follows:

bufcache (bufcache)
: Percentage of memory that the kernel wires for the metadata buffer cache. Increasing the value of the bufcache attribute can improve I/O performance by providing more memory for caching UFS file system data. Increasing the value of the bufcache attribute can free memory resources. For systems that only use AdvFS, you may want to decrease the value to 1 percent.

  Default value: 3 (percent) for 32-MB or larger systems; 2 (percent) for 24-MB systems

buffer-hash-size (bufhsz)
: Size of the hash chain table for the metadata buffer cache. The hash chain table is used to store the heads of the hashed buffer queues. A large hash chain table distributes the buffers and may make chain lengths short. This can reduce linear searches and improve lookup speed.

  Default value: 512 (slots)

fifo-do-adaptive
: When set to 0 (zero), the attribute disables the pipe code that attempts to batch writes to a pipe and then deliver this data in a single call to a reader.

  Default value: 1 (enabled)

* max-free-file-structures (max_free_files)
: Maximum number of file structures on the free list. When the number of free file structures that are chained for reuse on the free list reaches the value of the max_free_files attribute, file structure deallocation begins.

  You can modify this attribute at run time.

  Default value: 0

* max-ufs-mounts (max_ufsmounts)
: Maximum number of UFS or MFS file system mounts. You can increase the value of the max-ufs-mounts attribute if you want to

mount more than the default number of UFS or MFS file systems. This attribute does not affect performance.

You can modify this attribute at run time.

Default value: 1000

***** `max-vnodes` (`max_vnodes`)

Maximum number of vnodes (open files) on a system. The minimum value is specified by the `nvnode` attribute. The maximum value is the number of vnodes that 5 percent of the available memory can contain.

Increasing the value of the `max-vnodes` attribute allows more vnodes on a system, which may improve performance if your applications or users create a large number of open files. However, supporting more vnodes uses additional memory.

You can modify this attribute at run time.

Default value: 1000 (for 24-MB systems); the maximum value (for 32-MB or larger systems)

***** `min-free-vnodes` (`min_free_vnodes`)

Minimum number of free vnodes on the free list. If the number of vnodes on the free list is less than the value of the `min-free-vnodes` attribute, vnodes are deallocated.

Increasing the value causes the system to cache more free vnodes and may improve performance for vnode cache lookup operations. However, a large value increases the demand for memory.

You can modify this attribute at run time.

Default value: `nvnode` attribute (for 32-MB or larger systems); 150 (for 24-MB systems)

`name-cache-hash-size` (`nchsz`)

Size of the hash chain table for the namei cache. Large hash chain tables distribute the namei cache elements and may make chain lengths short, which can reduce linear searches and improve lookup speeds. In general, chains should contain only 2 or 3 elements.

Maximum value: `name-cache-size`/2

Default value: 256 slots

`name-cache-size` (`nchsize`)

Number of elements in the namei cache. Increasing the value of the `name-cache-size` attribute may improve lookup speeds, but it requires more memory. Decreasing the value can free memory. The maximum value is 2*`max-vnodes`*11/10.

Default value: 2*nvnode*11/10 (for 32-MB or larger systems); 150 (for 24-MB systems)

namei-cache-valid-time (ncache_valid_time)

Amount of time a namei cache entry can remain in the cache before it is discarded. A large namei-cache-valid-time attribute value will retain more vnodes references in the namei cache and improve the namei cache lookup speed; however, it will require more memory resources. A small value may cause premature deallocation of vnodes and decrease the namei cache lookup speed.

Default value: 1200 (seconds) for 32-MB or larger systems; 30 (seconds) for 24-MB systems

noadd-exec-access

Allows you to configure your system so that new executables cannot be created. This is a security feature and does not affect performance.

nvnode (nvnode)

Obsolete. Retained only for compatibility purposes. Determines the maximum and the minimum number of vnodes on a system. If you modify the value of the nvnode attribute and then reboot the system, the lockfile zone size and the specinfo_zone size allocation are also modified.

Default value: nproc+(2*maxusers)+128

path-num-max (path_num_max)

Size of the pathname zone for pathname lookup buffers. Increasing the value of path-num-max increases the number of elements in the zone that is allocated for pathnames.

Default value: 64 (zone elements)

pipe-maxbuf-siz

Maximum number of bytes buffered per pipe.

Default value: 65536 (64 KB)

pipe-single-write-max

Maximum size of a single write to a pipe.

Default value: -1

pipe-databuf-size

Number of data bytes in each pipe data buffer.

Default value: 8192 (bytes)

`pipe-max-bytes-all-pipes`
: Maximum number of bytes reserved for all pipes.

  Default value: 819200 (bytes)

`special-vnode-alias-tbl-size` (`spechsz`)
: Size of the special vnodes alias table for vnodes of special device files, such as character-I/O or block-I/O device files. Increasing the table size allows you to create more special device files.

  Default value: 64

`sys-v-mode` (`sys_v_mode`)
: Obsolete System V attribute.

`ucred-max` (`ucred_max`)
: Obsolete.

***** `vnode-age` (`vnode_age`)
: Amount of time that a vnode can remain on the free list before it is deallocated. You can increase the value of the `vnode-age` attribute to keep vnodes on the free list longer, which increases the possibility that the vnode will be successfully looked up.

  You can modify this attribute at run time.

  Default value: 120 (seconds) for 32-MB or larger systems; 2 (seconds) for 24-MB systems

`vnode-deallocation-enable`
: Enables (1) or disables (0) vnode deallocation. Enabling vnode deallocation decreases memory usage because it returns to the system the memory allocated to vnodes.

  Default value: 1 (enabled)

B.21 Virtual Memory Subsystem Attributes

The virtual memory (`vm`) subsystem attributes are as follows:

`contig-malloc-percent`
: Percentage of physical memory that is reserved for contiguous physical memory allocation at boot time.

  Default value: 20 (percent)

`dump-user-pte-pages`
: Determines if the pages from the user page table are written as part of a crash dump. Enabling this functionality provides more debugging information when a system crashes.

  Default value: 0 (disabled)

`gh-chunks`
: Number of 4-MB chunks of memory reserved at boot time for shared memory use.

`gh-min-seg-size`
: Size of a shared memory segment at which shared memory is allocated from the memory reserved for shared memory, according to the value of the `gh-chunks` attribute.

  Default value = 8 (MB)

`gh-fail-if-no-mem`
: When enabled (1), causes the `shmget` function to return a failure if the requested segment size is larger than the value of the `gh-min-seg-size` attribute, and if there is insufficient memory allocated by the `gh-chunks` attribute to satisfy the request.

  Default value = 1 (enabled)

`kernel-stack-guard-pages`
: When enabled, causes kernel stacks to be separated with unmapped guard pages. Guard pages are debugging aids that help isolate kernel stack corruption bugs caused by either overflowing or underflowing the kernel stack.

  Default value: 1 (enabled)

`new-wire-method`
: Internal use only. Do not modify.

  Default value: 0 (off)

`private-cache-percent`
: Specifies the percentage of the secondary cache that is reserved for anonymous (nonshared) memory. 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.

  Default value: 50 (percent)

`ubc-borrowpercent` (`ubc_borrowpercent`)

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

Default value: 20 (percent)

`ubc-maxdirtywrites`

Number of I/O operations (per second) that the virtual memory subsystem performs when the number of dirty (modified) pages in the UBC exceeds the value of the `vm-ubcdirtypercent` attribute.

Default value: 5

`ubc-maxpercent` (`ubc_maxpercent`)

Maximum percentage of physical memory that the UBC can use at one time.

Default value: 100 (percent)

`ubc-minpercent` (`ubc_minpercent`)

Minimum percentage of physical memory that the UBC can use.

Default value: 10 (percent)

`vm-aggressive-swap`

When enabled, causes the task swapper to aggressively swap out idle tasks, which prevents a low-memory condition from occurring. Aggressive task swapping allows more jobs to be run simultaneously, but it may decrease the interactive response time on a system that is excessively paging and swapping.

Default value: 0 (disabled)

`vm-asyncswapbuffers` (`asyncswapbuffers`)

Number of asynchronous I/O requests per swap partition that can be outstanding at one time. Asynchronous swap requests are used for pageout operations and for prewriting modified pages.

Default value: 4

`vm-clustermap`(`clustermap`)

Size of the kernel cluster submap, which is used to allocate the scatter/gather map for clustered file and swap I/O.

Default value: 1048576 (1 MB)

`vm-clustersize` (`clustersize`)
: Maximum size of a single scatter/gather map for a clustered I/O request.

  Default value: 65536 (64 KB)

`vm-cowfaults` (`cowfaults`)
: Number of times that the pages of an anonymous object are copy-on-write faulted after a fork operation but before they are copied as part of the fork operation.

  Default value: 4

`vm-csubmapsize` (`csubmapsize`)
: Size of the kernel copy submap.

  Default value: 1048576 (1 MB)

`vm-heappercent` (`heappercent`)
: Percentage of physical memory allocated to the kernel heap. Many of the kernel data structures are allocated from the kernel heap. The kernel heap wires physical memory as the kernel data structures are allocated.

  Default value: 7 (percent)

`vm-inswappedmin` (`inswappedmin`)
: Minimum amount of time, in seconds, that a task remains in the inswapped state before it is considered a candidate for outswapping.

  Default value: 1 (second)

`vm-kentry_zone_size` (`kentry_zone_size`)
: Size of the kernel's map entry zone submap. Kernel map entries are reserved for both pageable and nonpageable regions of kernel virtual address space.

  Default value: 16777216 (16 MB)

`vm-mapentries` (`mapentries`)
: Maximum number of map entries that any process can use at one time. Each map entry describes one unique disjoint portion of a virtual address space.

  Default value: 200

`vm-map-index-enabled`

Controls whether map entries are fully indexed. Each map entry describes one unique disjoint portion of a virtual address space. If set to 1 (enabled), map entries are indexed in all processes.

Default value: 1 (enabled)

`vm-map-index-count`

Controls the size of the map entry index. A large index can improve the lookup time, but may affect index rebalancing.

Default value: 64

`vm-map-index-rebalance`

Controls how frequently the map entry index gets rebalanced. If the difference between the longest map entry list and the shortest map entry list is greater than the value of the `vm-map-index-rebalance` attribute, the system will rebalance the index.

You can decrease the value of the `vm-map-index-rebalance` to improve the lookup time. However, this will increase the rate of rebalancing. If you increase the value of the `vm-map-index-rebalance` attribute, you will experience less rebalancing but have slower lookups.

Default value: 128

`vm-map-index-hiwat`

Controls when the system creates a map entry index. When a process allocates map entries equal to the value of the `vm-map-index-count` attribute multiplied by the `vm-map-index-hiwat` attribute, the system creates a map entry index for fast lookups.

Default value: 4

`vm-map-index-lowat`

Controls when the system deletes a map entry index. When a process removes enough map entries from an index so that the number of entries is less than the value of the `vm-map-index-count` attribute multiplied by the value of the `vm-map-index-lowat` attribute, the system deletes the map entry index.

Default value: 2

`vm-max-rdpgio-kluster` (`vm_max_rdpgio_kluster`)

Size of the largest pagein (read) cluster that is passed to the swap device.

Default value: 16384 (16 KB)

`vm-maxvas` (`maxvas`)
: Maximum amount of virtual address space that a user process can use at one time.

  Default value: 1073741824 (1 GB)

`vm-maxwire` (`maxwire`)
: Maximum amount of memory that any user process can wire. (Paging activity generally increases as the amount of wired memory increases.)

  Default value: 16777216 (16 MB)

`vm-max-wrpgio-kluster` (`vm_max_wrpgio_kluster`)
: Size of the largest pageout (write) cluster that is passed to the swap device.

  Default value: 32768 (32 KB)

`vm-min-kernel-address`
: Base address of the kernel's virtual address space. The value can be either Oxffffffff80000000 or Oxfffffffe00000000, which sets the size of the kernel's virtual address space to either 2 GB or 8 GB, respectively. You may need to increase the kernel's virtual address space on very large (VL) systems (for example, systems with several gigabytes of physical memory and several thousand large processes).

  Default value: 18446744071562067968 (2 to the power of 64)

`vm-page-free-min` (`vm_page_free_min`)
: Paging begins when the number of pages on the free page list falls below this value.

  Default value: 20 (pages)

`vm-page-free-optimal` (`vm_page_free_optimal`)
: Hard swapping begins when the number of pages on the free page list falls below this value for five seconds.

  Default value: 72 (pages)

`vm-page-free-reserved` (`vm_page_free_reserved`)
: Only privileged tasks can get memory when the number of pages on the free page list falls below this value.

  Default value: 10 (pages)

`vm-page-free-target` (`vm_page_free_target`)
Paging stops when the number of pages on the free page list reaches this value.

Default value: 128 (pages)

`vm-page-free-hardswap`
Task swapping stops when the number of pages on the free page list reaches this value.

Default value: 1280 (pages)

`vm-page-free-swap`
Idle task swapping begins when the number of pages on the free page list falls below this value.

Default value: 74 (pages)

`vm-page-lock-count`
Size of the lock array that is used to synchronize access to `vm_page` kernel structures. Instead of locking each page structure, the virtual address is used to hash into the lock array. Adjust this value only if excessive lock contention occurs.

Default value: 64

`vm-page-prewrite-target`
Maximum number of pages that the `vm` subsystem will prewrite to swap space if it anticipates running out of memory. The prewritten pages are the least recently referenced (LRU) pages.

Default value: 256 (pages)

`vm-segmentation` (`segmentation`)
When enabled, causes shared regions of user address space to share the page tables that map to those shared regions.

Default value: 1 (enabled)

`vm-segment-cache-max` (`u_seg_cache_max`)
Number of text segments that can be cached in the segment cache. (Applies only if you enable segmentation.)

The `vm` subsystem uses the segment cache to cache inactive executables and shared libraries. Because objects in the segment cache can be accessed by mapping a page table entry, it eliminates I/O delays for repeated executions and reloads.

Reducing the number of segments in the segment cache can free memory and help to reduce paging overhead. (The size of each segment depends on the text size of the executable or the shared library that is being cached.)

Default value: 50 (segments)

`vm-syncswapbuffers` (`syncswapbuffers`)

Number of synchronous I/O requests that can be outstanding to the swap partitions at one time. Synchronous swap requests are used for pagein operations and task swapping.

Default value: 128

`vm-syswiredpercent` (`syswiredpercent`)

Maximum percentage of physical memory that can be dynamically wired. The kernel and user processes use this memory for dynamically allocated data structures and address space, respectively.

Default value: 80 (percent)

`vm-ubcbuffers` (`ubcbuffers`)

Total number of UBC I/O requests that can be outstanding at one time.

Default value: 256

`vm-ubcdirtypercent` (`ubcdirtypercent`)

The UBC starts writing dirty (modified) pages when the number of dirty pages reaches this value.

Default value: 10 (percent)

`vm-ubcpagesteal`(`ubcpagesteal`)

The UBC steals file pages to satisfy the file's demand for pages when the number of pages for a file falls below this value.

Default value: 24 (file pages)

`vm-ubcseqpercent` (`ubcseqpercent`)

Specifies the maximum amount of UBC memory that can be used to cache a single file.

Default value: 10 (percent)

`vm-ubcseqstartpercent` (`ubcseqstartpercent`)

The UBC starts recognizing sequential file access and stealing the UBC LRU pages for a file to satisfy its demand for pages when the size of the UBC reaches this percentage of physical memory.

Default value: 50 (percent)

`vm-vpagemax` (`vpagemax`)

Maximum number of virtual pages within the address space for a process that can be given individual protection attributes (that is, protection attributes that differ from the protection attributes associated with the other pages in the address space).

Changing the protection attributes of a single page within a virtual memory region causes all pages within that region to be treated as though they had individual protection attributes. For example, each thread of a multithreaded task has a user stack in the stack region for the process in which they run. Because multithreaded tasks have guard pages (that is, pages that do not have read/write access) inserted between the user stacks for the threads, all pages in the stack region for the process are treated as though they have individual protection attributes.

If a stack region for a multithreaded task exceeds 16 KB pages, you may want to increase the value of the `vm-vpagemax` attribute.

Default value: 16384 (16 KB)

`vm-zone_size`(`zone_size`)

Percentage of physical memory that is allocated to the kernel's zone submap. Many of the dynamically allocated kernel data structures are allocated out of zones that are located in the zone submap.

Default value: 0 (percent)

B.22 XPR Subsystem Attributes

The XPR (`xpr`) subsystem attributes are as follows:

`nxprbufs` (`nxprbufs`)

Number of contiguous buffers allocated for XPR. Each of these buffers is of type `struct xprbuf`, which occupies 56 bytes. The buffers are not allocated if XPR is not turned on.

XPR is a facility that performs silent tracing using a circular buffer. XPR logs the pointer to a `printf` string and up to six arguments, along with a timestamp and CPU information (for multiprocessor systems), into a circular buffer.

The kernel compile-time flag `XPR_DEBUG` determines whether the XPR debugging facility is included in compilations. By default, the XPR facility is not included in compilations.

Default value: 0

`xprflags` (`xprflags`)

Bit mask in which each bit controls a message type. A type of message is enabled or disabled by setting the value of the corresponding bit in the `xprflags` mask to 1 or 0.

The mapping of the bits in the `xprflags` mask and the different message types are as follows:

Bit Position	**Message Type**
0	XPR_SYSCALLS
1	XPR_TRAPS
2	XPR_SCHED
5	XPR_TCP
6	XPR_PMAP
7	XPR_VM_MAP
8	XPR_VM_OBJECT
9	XPR_VM_OBJECT_CACHE
10	XPR_VM_PAGE
11	XPR_VM_PAGEOUT
12	XPR_MEMORY_OBJECT
13	XPR_VM_FAULT
14	XPR_INODE_PAGER
15	XPR_INODE_PAGER_DATA
16	XPR_TTY
17	XPR_BIO
18	XPR_INTR
19	XPR_CACHE
20	XPR_NFS
21	XPR_SIGNAL

See the definition of the `nxprbufs` attribute for more information about the XPR facility.

Glossary

This glossary lists the terms that are used to describe performance and availability.

active list

Pages that are being used by the virtual memory subsystem or the UBC.

adaptive RAID 3/5

Also called dynamic parity RAID, adaptive RAID 3/5 functionality improves disk I/O performance for a wide variety of applications by dynamically adjusting, according to workload needs, between data transfer-intensive algorithms and I/O operation-intensive algorithms.

anonymous memory

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.

eager mode

See **immediate mode**.

fail over

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.

lazy mode

See **deferred mode**.

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.

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 DIGITAL 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 data 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 2*n* 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 by the kernel or it is shared by virtual memory 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).

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 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.

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

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 by the kernel and cannot be reclaimed.

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

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-DIGITAL (800-344-4825) before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-234-1998 using a modem from anywhere in the USA, Canada, or Puerto Rico. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	—	Local Digital subsidiary or approved distributor
Internal (submit an Internal Software Order Form, EN-01740-07)	—	SSB Order Processing – NQO/V19 *or* U.S. Software Supply Business Digital Equipment Corporation 10 Cotton Road Nashua, NH 03063-1260

Reader’s Comments

DIGITAL UNIX
System Configuration and Tuning
AA-Q0R3F-TE

Digital 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 ---------------------------------------

digital™

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

DIGITAL EQUIPMENT CORPORATION
UBPG PUBLICATIONS MANAGER
ZKO3 3/Y32
110 SPIT BROOK RD
NASHUA NH 03062 9987

-------- Do Not Cut or Tear - Fold Here --

Cut on Dotted Line