

Sun StorEdge 5000 Family Software

A Technical White Paper
December 2004



© 2004 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95054 USA

All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California.

Sun, Sun Microsystems, the Sun logo, Solaris, and Sun StorEdge are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a). DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS HELD TO BE LEGALLY INVALID.



Please
Recycle



Adobe PostScript

Table of Contents

Introduction	1
File System Architecture	3
Metadata and Data	4
Journaling	4
Importance of Journaling and Metadata	5
Why Journal on Disk	5
Expanding Volumes	5
LUN	6
Partitions	6
File Volumes	6
Segments	6
Checkpoints	9
Overview	9
Using Checkpoints	10
How Checkpoints Work	10
An In-Depth Look at Checkpoints	12
Volumes	12
Checkpoint Lifecycle	13
Creating Checkpoints	13
Active Checkpoints	13
Translating File System Objects in Checkpoints	15
Deleting Checkpoints	15
Scheduling Checkpoints	17
Accessing Checkpoints	17
Compatibility Issues	18
Restoring Objects from Checkpoints	18
The cp Command	19
Network Data Sharing	21
UNIX® NFS Security Model	21
Windows CIFS Security Model	22

Connection-Oriented vs. Connectionless	22
File System Features for File Sharing	22
Windows Security and Access Control Lists	23
CIFS Security Policy	23
Mapping a Windows ACL to UNIX Permissions	24
Mapping UNIX Permissions to Windows ACL	25
DOS Attributes	26
Cross-Platform File Locking	26
File Locking Caveats	27
Managing Space	27
Unicode Support	27
Shares and Support for ADS and Dynamic DNS	29
Shares	29
Autohome Shares	30
Support for Active Directory Service (ADS) and Dynamic DNS	30
Configuring ADS	31
Configuring Dynamic DNS	31
Publishing SMB/CIFS Shares in ADS	32
Conclusion	33
References	35
Sun Web Sites	35
Glossary	37

Chapter 1

Introduction

Today, more and more users need the ability to access and share files between Windows and UNIX® (including Linux) systems, but sharing files in a heterogeneous environment is a challenging task. One solution is to install NFS on PC clients, but this can be expensive, time-consuming, and is not PC-friendly. Another method is to employ separate file servers for each protocol, which increases administrative complexity and results in redundant investments in storage. This method also duplicates backup and restore functions and still presents an impediment to applications that need to share data between UNIX and PC users. In addition to these issues, the increasing reliance on file servers for user and application data is driving a requirement for higher reliability, performance, availability, and the ability to quickly restore files.

The Sun StorEdge 5000 family of NAS appliances is specifically designed to provide an ideal storage solution for multi-protocol networks by offering a single system that can contain both UNIX and Windows files. The Sun StorEdge 5000 family's operating system is intended for different requirements than those of general operating systems. It is developed for interoperability with Network File System (NFS) and Common Internet File System (CIFS) architectures, as well as increased reliability, decreased downtime, ease of use and manageability, and high performance. Most of these characteristics are highly dependent on the features of the family's innovative file system. These features include:

- Journaling both metadata and data for higher reliability, availability, and the ability to create large file systems
- Dynamic volume expansion to increase capacity on-line
- Checkpointing for easier and faster backups and restores
- Secure network data sharing

This paper provides an in-depth review of the Sun StorEdge 5000 family software starting with the file system architecture in Chapter 2, which includes discussions on metadata and data journaling and dynamic volume expansion. Chapter 3 contains an overview of checkpoints followed by a deeper examination into exactly how checkpoints function. Network data sharing is the topic detailed in Chapter 4, highlighting the difference between the UNIX and Windows security models and how the file system addresses them, as well as file locking. And, Chapter 5 provides information on how Windows users access files via shares and how to administer shares in directory and domain services.

Chapter 2

File System Architecture

The Sun StorEdge 5000 family is designed with a 64-bit file system, allowing a maximum address space of up to 16 exabytes. Given this theoretical upper bound, it is possible for a single file system to scale up to 16 TB. A single Sun StorEdge 5000 family NAS appliance can support up to 512 volumes, or file systems, for a maximum hypothetical capacity of 8 petabytes. The scalability of the file system positions it well into the future in terms of supporting larger file systems, larger files, and larger disk drives.

The file system is a block-based file system with 4 KB block sizes. As with most UNIX based file systems, it utilizes inodes — data structures containing information about files. Each file is associated with a uniquely identified inode in the file system in which it resides. Each inodes contains the mode and type of the file, the number of links to the file, the owner's user and group IDs, the number of bytes in the file, access and modification times, the time the inode itself was last modified, and the addresses of the files block on disk, as shown in Figure 1.

Figure 1. Content of inodes

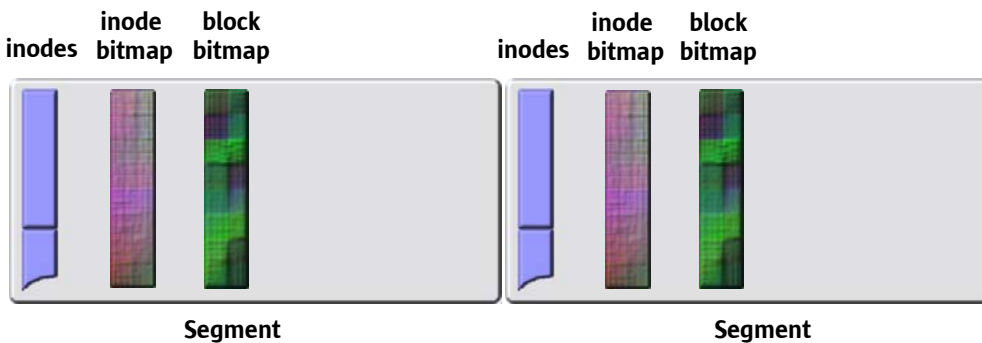
.inodes



- Mode and Type
- Number of Links
- Owner's UID and GID
- Number of Bytes
- Addresses of File Blocks
- Access and Modification Times

In order to map out free blocks on the file system, bitmaps are used to identify available blocks for the file system to use. For every block on the disk, there is a corresponding entry in the bitmap entry on the inode bitmap. All of the inode table entries, and inode and block bitmaps are located at the beginning of a file system segment (a volume of storage space that can be attached to an existing file volume at any time).

Figure 2. Each segment contains its own inodes, inode bitmap, and block bitmap



The Sun StorEdge 5000 family uses segments for scalability. Each segment of the file system contains its own inodes, inode bitmap, and block bitmap, illustrated in Figure 2. Segments can be added to increase the size of the file system while the NAS appliance is running and in use.

Metadata and Data

File systems contain two types of data — 1) regular file data that is stored in the file system and 2) metadata, which contains the inode and bitmap entries that store information about the organization of a file system's file data. Metadata is responsible for keeping track of the information about the files in the file system. When the content of a file changes, some of the related metadata changes as well. For example, when a file is deleted, the blocks that belonged to the file are released to the file system for other use. Therefore, a file system operation such as writing to a file or deleting a file can affect more than one part of the file system and consist of several operations on disk. In order to keep the file system consistent, it is imperative that either all of the operations complete because if only some of the operations complete when a file is deleted, the file system would be left in an inconsistent state.

The consistency of a file system can be jeopardized when a system or power failure occurs while a file system operation is under way. To address this issue, some operating systems use file system-specific software, such as *fsck* and *scandisk*, to help ensure the consistency of the file system. These utilities scan the file system for errors and any unassociated links. This type of process can take hours on large file systems, making it impractical for the highly scalable file systems available on the Sun StorEdge 5000 family. To overcome this problem, the Sun StorEdge 5000 family is designed with a journaling file system.

Journaling

In the event of a system crash or other failure, the integrity of the data on a file system can be compromised. The file system's journaling feature maintains a log, or journal, of the block-level activity that occurs on the disk. If there is a failure, any lost data can be recreated because updates to the metadata in directories and bitmaps are written to the journal.

The journaling file system not only returns the data to a pre-crash state, but also recovers any unsaved data and stores it in the location it would have been stored to if the operation had not been unexpectedly interrupted. With the journaling file system, the Sun StorEdge 5000 family can recover from an unexpected shutdown in a few minutes, as opposed to requiring a costly, time-consuming consistency check.

Importance of Journaling and Metadata

Most journaling file systems only journal the metadata changes. This helps ensure file system consistency after a crash because all of the metadata updates are in the journal and can be applied to the file system to recreate previously committed transactions. However, since only the metadata, and not the data, is journaled, any changes to the data itself are lost. To mitigate this problem, the higher layer applications (such as NFS and CIFS) are designed to acknowledge transactions to the client only after the data is written to its permanent position on the disk. This synchronous write method increases the response time of the file system, thus degrading the performance of write operations.

The Sun StorEdge 5000 family's journaling file system solves both the data integrity and write performance problems by saving both data and metadata in the journal. The write performance issue is negated because the client is notified as soon as data is written to the journal. And, data integrity is preserved because the system commits all of the journal entries to their permanent locations when it recovers, so data is not lost if the system crashes or fails.

Why Journal on Disk

The file system utilizes an on-disk journal with very impressive performance attributed to the small contiguous area on disk that is allocated for it. This contiguous area helps ensure that all writes to the journal are sequential and minimize seek time, of which the greatest factor is disk access latency. Since the client is notified of all writes as soon as they are in the journal, the client's perceived write latency is significantly less than it is without data journaling.

An alternative to storing the journal on disk is to store it in non-volatile RAM (NVRAM). Although the performance of NVRAM is better than disk, there are several drawbacks to this approach. First, NVRAM implementations are very costly due to the cost of NVRAM itself and due to the complex, proprietary hardware required to operate it. Second, NVRAM is solely dependent on the availability of power to keep the data within it — if power is lost, NVRAM must rely on a battery backup. This dependency introduces a single-point-of-failure for important data when the NVRAM battery is in use. Finally, NVRAM journaling requires elements of the file system to be shared between two devices — disk and NVRAM. This creates an obstacle when sharing the state of the file system with another system is required. For example, clustered failover applications utilizing journaling on NVRAM require complex, expensive journal interconnects between cluster nodes that incur a penalty on overall system performance.

With the Sun StorEdge 5000 family, these drawbacks do not exist because the journal is located solely on RAID-protected hard disks. Since the journal is on highly available RAID drives, there is no dependency on a single-point-of-failure to maintain the journal information. And, sharing the journal between two nodes in a cluster configuration is possible using the standard Fibre Channel (FC) protocol, offering cost-effective high availability and access to shared storage systems.

Expanding Volumes

Scalability is a vital capability of any enterprise file server. The option to add storage to a file system without affecting the availability of data is imperative for today's business-critical applications that can not tolerate downtime. With the Sun StorEdge 5000 family, an existing file system can be extended in seconds while the array is on line and accepting data I/O. The Sun StorEdge 5000 family utilizes extension segments to extend the file system. Before explaining how this is accomplished, it is necessary to understand some of the concepts of the file system and how file volumes are constructed. The following sections describe LUNs, partitions, file volumes, and segments as they apply to the Sun StorEdge 5000 family of NAS appliances.

LUN

LUN stands for Logical Unit Number and identifies the logical representation of a physical or virtual device. In the Sun StorEdge 5000 family, there is a one-to-one correspondence between RAID sets and LUNs. However, LUNs are managed as separate entities and treated as a single storage volume. By treating LUNs this way, the Sun StorEdge 5000 family greatly simplifies the process of establishing a file system. The space on the RAID set is accessed independently of the physical drive limits through the LUN.

Partitions

Partitions are sections on a LUN that provide a way to subdivide the total space available within a LUN. The Sun StorEdge 5000 family operating system supports a maximum of 31 partitions per LUN. When a LUN is first created, all of the available space is located in the first partition and any others are empty. To use the space in a partition, a file volume must first be created. Each partition can contain only one file volume. However, a single file volume can span several partitions. When a file volume is created, the size of the partition is automatically adjusted to match the size of the file volume. Any additional space on the LUN is automatically assigned to the next partition. Once the maximum number of file volumes for the operating system are created, any extra space on that LUN is inaccessible.

File Volumes

A file volume defines the space that is available for storing information and is created from a partition with available space that is not already assigned to another file volume. If the volume does not use all of the available space in a partition, the remaining space is automatically allocated into the next partition. After four volumes are created on a LUN, any remaining space on that LUN is inaccessible. New file volumes are limited to 255 GB in size.

The size of a file volume can be increased by attaching a segment. The segment is essentially another file volume with special characteristics. When a segment is added to an existing volume, the two become inseparable and the user sees only one volume with more space. The flexibility of this system allows file volumes to be created and then expanded as needed without disturbing users or forcing them to spread their data over several volumes. Larger file volumes are created by attaching up to 62 segments to the original file volume.

This is all transparent to end users, who only see a single volume. If the file volume begins to fill up, the administrator can attach another segment and increase the available space within that file volume. In physical terms, this may involve adding more drives, RAID sets, even an entire NAS array. However, the physical aspect is invisible to a user: All the user sees is more storage space within the volume.

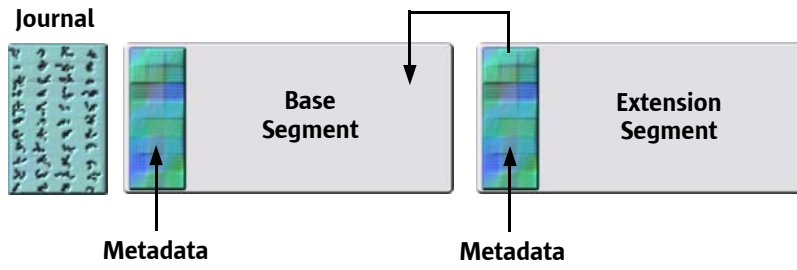
Segments

Segments are volumes of storage space created much like file volumes and they can be attached to an existing file volume at any time. Attaching a segment increases the original file volume's total capacity. Each segment must be created independently and then attached to a file volume. Once attached to the volume, the volume and segment are inseparable.

In general, segments are created as needed and attached to a volume as the volumes begin to fill with data. The main advantage of adding space by attaching segments is that a segment can be created on a new drive, or even on a new array, and once attached to the original file volume, the different physical storage locations are invisible to the user. Therefore, space can be added when it is needed, without bringing down the network or file systems to restructure the data storage and create a bigger file volume.

In terms of the journal and metadata, the journal is shared between the segments, but each segment contains its own metadata (block and inode bitmaps and the inode table), as illustrated in Figure 3 below.

Figure 3. The journal is shared between segments



Chapter 3

Checkpoints

Today, high availability is a business necessity, challenging system administrators with the need to preserve data integrity and provide round-the clock data availability. In a global economy, where speed is of the essence in order to stay competitive, traditional backup methods are increasingly incapable of retrieving files fast enough for users. For example, a CAD designer creates a complex architectural plan for a building, only to realize that the last save operation deleted the design for the basement. The save is already committed to the disk and the changes are now unrecoverable. The designers only recourse is to call the system administrator and ask them to restore a tape backup from the previous night. The designer may be able to recover a restored copy of the work, but it typically takes several hours, requiring the administrator to extract the data from a slow tape drive after they have found the exact file in the backup archive. In addition, the designer must re-create all of the work performed since the last backup. In this example, the high cost of lost productivity, both for the designer and the administrator illustrates a business case for using checkpoints.

The Sun StorEdge 5000 family is designed with an integrated checkpoint function that offers the ability to perform on-line volume backups and fast file recovery to help meet the challenging demands of today's users. A more favorable alternative to the traditional file backups in the example above is checkpointing, a feature that enables easy and fast restore of accidentally deleted, corrupted, or modified files.

Overview

A checkpoint is a point-in-time image of a Sun StorEdge 5000 family file system that is copied at the volume level. While the active file volume can be modified with read/write operations, the virtual volume that is produced when the checkpoint is created is available in a static, read-only state. The checkpoint is a virtual, imaginary image of the file volume. A checkpoint does not provide an online volume backup, as does a volume-duplicating tape backup. Instead, it contains pointers to all of the locations of data on the volume at the time the checkpoint is created.

The checkpoint operation itself is extremely fast and users typically do not notice when they are created. Consistent with the multi-protocol design of the Sun StorEdge 5000 family operating system, checkpoints can be accessed by both NFS and CIFS clients. The operating system maintains up to a maximum of 16 concurrent checkpoints. Therefore, with an eight hour workday, this enables a minimum of two days of hourly checkpoints for any point in time. Now, the designer from the example above can quickly restore the file from the checkpoint and continue working with the file as it existed at most 59 minutes in the past.

Using Checkpoints

As file systems grow larger, the ability to take snapshots of data at particular points in time enables shorter backup windows and faster restores of data. Specifically, checkpoints are ideal for:

- **Rapid file recovery** — Checkpoints can be used to quickly access and restore a recent image of a file that is modified, deleted, or corrupted accidentally or intentionally. With checkpoints, the old data still resides on disk, so users do not need to involve an administrator to restore previous versions of files.
- **Online backup of files** — Checkpoints enable administrators to backup large file systems without worrying that data is changing during the backup process. They simply create a checkpoint at the time they need to perform the backup and then run the backup process against the checkpoint. This helps ensure that the file system is always backed up in a consistent state and dramatically reduces backup windows.
- **Database backups** — Checkpoints can help make database applications more available by shortening the amount of time a database needs to be offline in order to perform a backup. The administrator pauses the database only long enough to create a checkpoint of the database at that point in time, which is then used to perform the backup at the administrator's convenience.

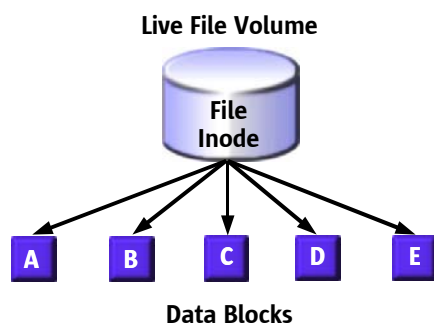
Some of the benefits of using checkpoints are:

- **Reducing recovery time** — With checkpoints, the time to recover data can be significantly reduced. Users no longer wait hours to retrieve day- or week-old files. The files are available directly from the file system itself.
- **Increasing user and administrator productivity** — Without checkpoints, both users and administrators are faced with the time-consuming tasks of recovering files and re-creating lost work. Checkpoints can reduce these issues because files are restored very quickly and in a more recent state.
- **Reducing cost** — Improving file restore times can translate into a large cost advantage in terms of increased productivity and decreased administrative overhead. In addition, because checkpoints simply contain pointers rather than a complete copy of a volume, they consume much less disk space than a full volume copy (depending on the rate of change of the files in the file volume and the number of checkpoints).

How Checkpoints Work

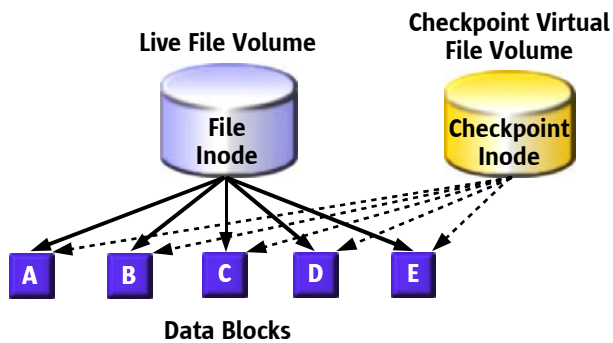
As previously discussed, the file system uses inodes to store specific file-related information, including the addresses of data blocks. The file block addresses are crucial to understanding how checkpoints operate. A live file volume contains inodes that describe, among other things, the addresses of the blocks that compose that particular file volume. This addressing is accomplished by using pointers to reference the data blocks that are stored elsewhere on the disk, as illustrated in Figure 4.

Figure 4. File volume before a checkpoint is created



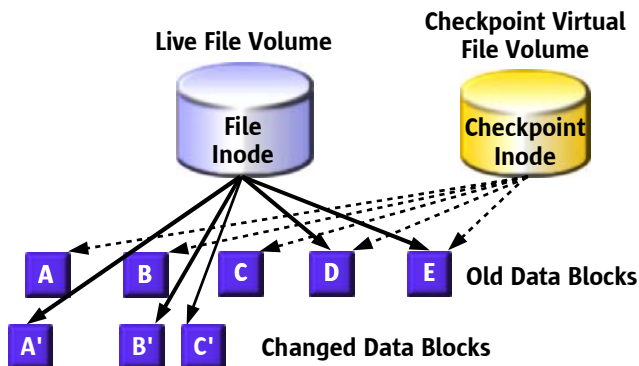
The file system creates a checkpoint by maintaining all pointers to disk blocks that are currently in use at the time of the checkpoint. As depicted in Figure 5, when the checkpoint is created, a new inode is produced to maintain existing pointers to the data blocks.

Figure 5. File volume after checkpoint is created



After the checkpoint is created, any changes to the files in the live file volume result in updates to the current set of pointers in the live file volume. Figure 6 shows the state of the volume after some of the data blocks are updated.

Figure 6. New data blocks are created when files are updated



This process is similar to the process that occurs when changes are updated on non-checkpointed volumes, which also update pointers to point to the new disk blocks. Because the process adheres to the standard volume update processes, performance degradation with checkpoints is limited. The actual impact on performance varies depending on a number of factors, including file system size, number of files, number of delete operations, depth of directories, average file size, and the outstanding number of checkpoints.

An In-Depth Look at Checkpoints

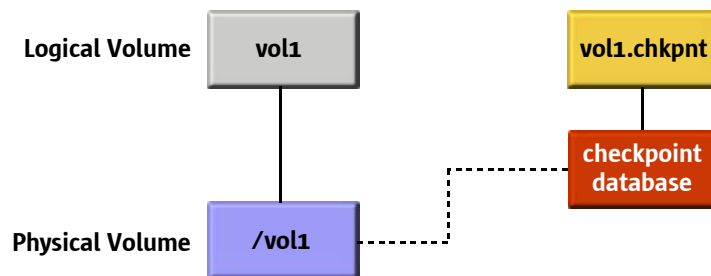
This section provides an in-depth look at how checkpoints are created, maintained, and deleted. The lifecycle of a checkpoint is divided into three main stages: creation, active pseudo file system, and deletion. These concepts are discussed below. But first, it is important to understand the relationship between the live file volume and the checkpoint.

Volumes

In the Sun StorEdge 5000 family operating system, every mounted file system is represented by an in-memory data structure called *fs_online*. The *fs_online* of a volume is similar to a gate — all accesses to the volume are routed from there. *fs_online* stores information about a volume, including the capacity, file-handle of the root directory, and status flags. A file-handle is the virtual identifier required for accessing any file system object in the file system. A file-handle maintains information about the virtual volume and the corresponding object in that volume. The file-handles for checkpoint volumes also contain information about the checkpoint identifier that contains the actual checkpointed object.

Checkpoints of a volume are accessed through a separate *fs_online*. This volume corresponds to the virtual checkpoint volume created when checkpoints are enabled on a volume. The existence of a checkpoint database distinguishes the checkpoint volume or CFS (Checkpoint File System) from the main volume or LFS (Live File System). The checkpoint database is the data structure that virtually holds different versions of an LFS. In essence, it is a mapping function that maps a virtual block address to its corresponding real address on the live file system. The relationship of the physical and logical volumes is illustrated in Figure 7.

Figure 7. Relationship between physical and logical volumes



The checkpoint database is a flat, sparse file with one entry per each block address in the live file system. Each entry is an array of 16 block addresses. When the mapping function tries to resolve a virtual block address, it first locates the corresponding entry in the checkpoint database. In order to do this, it uses the virtual block address as an index to the file. It then uses the checkpoint identifier as the index to the entry array to find the proper real block address.

Checkpoint Lifecycle

Checkpoints have three states:

- Active — While checkpoints are active they can be accessed for most read-only file system operations.
- Delete pending — When a checkpoint expires and is automatically removed by the system or explicitly removed by users, it is marked as *delete pending*. Later, one of the file system threads called *checkpoint cleaner* automatically removes it. When a checkpoint status is changed from active, it is no longer accessible and a new checkpoint with the same name can be created.
- Deleting — As the checkpoint cleaner is removing a checkpoint, its status is *deleting*.

Creating Checkpoints

There is a special mode page for checkpointing that is similar to a directory and contains all of the information about checkpoints for a file system. The address of this page is in the volume label of the file system and is allocated when checkpoints are active on a file system. This page contains a table of active checkpoints on the file system, properties of checkpoints, and an array of pointers that are the first of a three-level indirection to pages containing the mappings for the LFS blocks. There is also a stack of active checkpoints. Each checkpoint that is created is pushed to the top of the stack. Checkpoints are created in the *checkpoint pseudo-directory*.

Checkpoints are accessed through pseudo-volumes named */VOL.chkpt*, where *VOL* is the name of the checkpoint volume. Under the top, or root, directory of a *VOL.chkpt* volume is a set of pseudo-directories, each with the name of a checkpoint.

These pseudo-directories hold the virtual images, or checkpoints, of the entire volume. Each directory is clearly labeled with a time stamp that describes the attributes (time it is created and how long it is to be retained) of each checkpoint. For example, a checkpoint directory labeled:

```
20040719-010001,7d/home/joe
```

is a checkpoint of the files and directories contained in */home/joe* on July 19, 2004 at 1:00 in the morning. This particular checkpoint is scheduled to be retained for seven days.

Active Checkpoints

Each CFS has a mapping function for all of the blocks on the LFS. This mapping function returns a value for each page of the LFS. The operating system on the Sun StorEdge 5000 family supports up to 16 active checkpoints per file system. This means the checkpoint file system can hold at most 16 mappings for each block. However, not all blocks in the LFS are mapped and checkpointed, i.e., the block allocation table and journaling area.

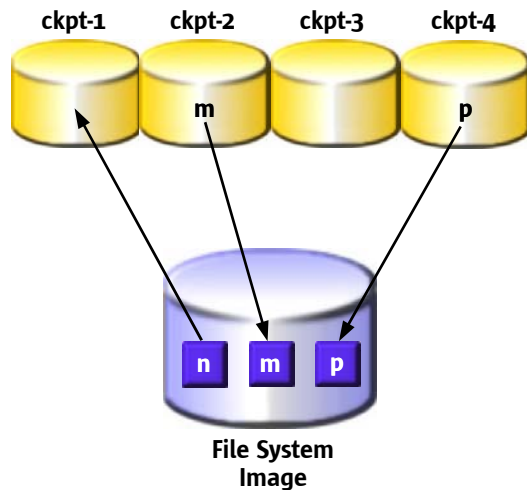
Active checkpoints have a stack-like structure and each checkpoint uses the mapping of itself. If a mapping does not exist, it uses the mapping for the next checkpoint created after it. If none of the entries have a mapping, this implies the file system block has not been modified since the oldest checkpoint was created. In this case, the block address is mapped to itself.

When a new checkpoint is created, a new directory entry is added to the list of checkpoints in the checkpoint control page. Also, an available slot (out of 16 slots) in the checkpoint stack is assigned to this newly-created checkpoint. After doing this, the new checkpoint becomes the owner of the corresponding entry in mapping entries for all blocks of the file system. The entire operation is performed in a single transaction and is instantaneous. As soon as a new checkpoint is created, the checkpoint entry becomes available and can be used for read-only file system operations. There is no partial state visible from other file system users.

Figure 8 depicts part of an array of mappings for block *n*. In this example, ckpt-4 is the most recent checkpoint and therefore the last entry in the mappings, and ckpt-1 is the oldest checkpoint. In this example, assume there is a file system operation on ckpt-1 and block *n* is accessed. The mapping function first checks the mapping for

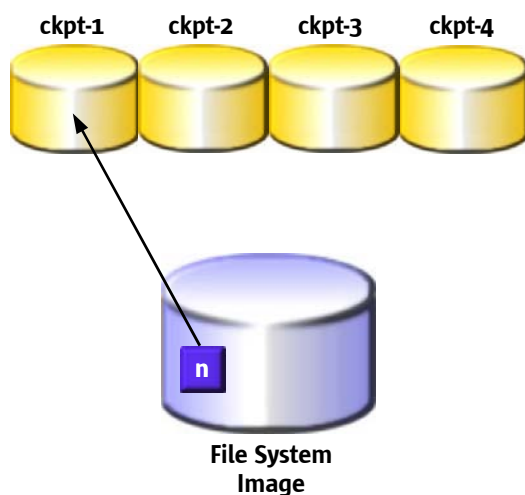
ckpt-1, and because it is empty, moves forward and checks the mapping for ckpt-2. It finds m and uses block m instead of block n . When accessing block n from ckpt-2, which is empty, the mapping function moves forward and uses the mapping for ckpt-3, which is block p . Note that when searching for a mapping entry, the system always moves forward from the current checkpoint toward more recently created checkpoints. The mapping entries of checkpoints that are older than the current checkpoint are never used.

Figure 8. Copy-on-write mechanism for checkpoints



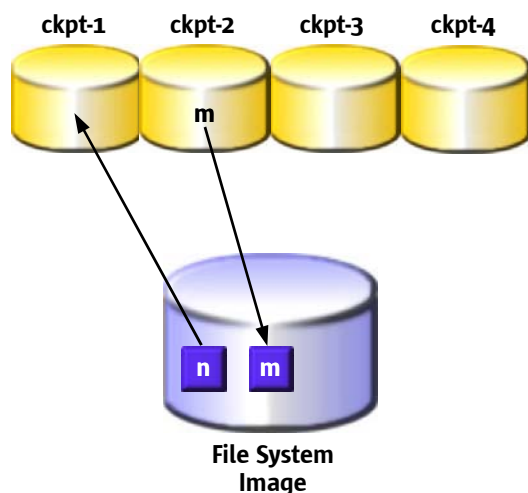
While checkpointing is active for a volume, all of the blocks of the LFS are shared by the checkpoint to its corresponding CFS until they are changed. As shown in Figure 9, while block n is not modified, all the checkpoints of the file system use the block n itself with no mapping. If a block in the live file system is modified, the update function first checks to see if there is an existing mapping for this block. If there is a mapping for this block, the function does nothing. If no mappings can be found and the change is not for block allocation, then it duplicates the block and puts the address of the new block into the mappings for the most recently created checkpoint. If the change is for block allocation, the system does not duplicate the block, it puts a special value into the mapping. If the block is newly allocated, there cannot be any object on the checkpoints using this block, and therefore it does not need to be duplicated.

Figure 9. Mapping for block n before it is modified



In Figure 10, there are two active checkpoints, ckpt-1 and ckpt-2. If block n is going to be modified, a new block m is allocated and the old content of n is copied to it. The address for block m is inserted into the list of mappings for the block n in the latest checkpoint, which is ckpt-2. From this point, any request for block n in ckpt-1 and ckpt-2 is redirected to the block m , where the original content resides.

Figure 10. Mapping for block n after it is modified



Referring back to Figure 8, ckpt-1 is created and then ckpt-2 is created before block n is modified. Therefore, ckpt-1 and ckpt-2 represent the same content for block n . Block n is then modified before ckpt-3 is created. A copy of the *before change content* is created and inserted into the mapping for the most recent checkpoint, which at the time is ckpt-2. Because both ckpt-1 and ckpt-2 expect to see the same content for block n , the mapping from n to m is shared by both of them.

In Figure 8 again, ckpt-3 is created and without modifying block n , ckpt-4 is created and then block n is modified again. This causes block p to be created and a mapping from n to p for checkpoint-3 and ckpt-4.

When deleting a block in the LFS, the system first checks if there is a mapping for the block in the most recent checkpoint. If there is a mapping, it frees the block. Otherwise, it puts the address of the block into the mapping for it, creating a one-to-one mapping for that block. This is called *page stealing* of the CFS from the LFS.

Translating File System Objects in Checkpoints

As previously discussed, because the checkpointing mechanism is applied to file system blocks rather than file system objects, there is no special consideration for the *type* of object that is checkpointed. Hard links and symbolic links in checkpoints continue to have the same semantics they had in the live file system at the time the checkpoint is created. To keep hardlinks unchanged, the directory entries referring to the same inode and the inode itself have the unchanged values. Also, checkpoints in all the disk blocks continue to have the old values, regardless of whether they are metadata or data blocks — all the hard links remain unchanged. Symbolic links are simply data files that are most probably references to other objects, therefore the same applies to them.

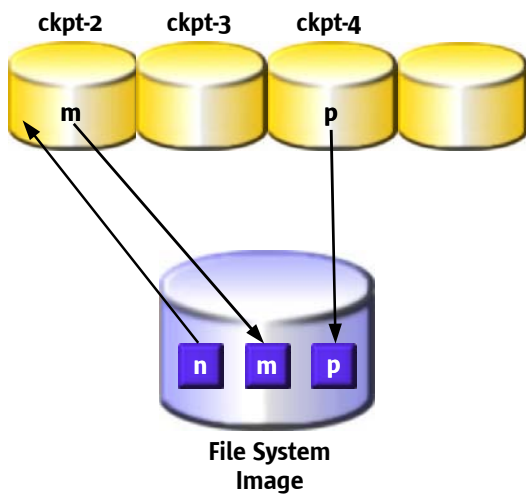
Deleting Checkpoints

When a checkpoint is to be deleted, a flag is set in the entry for that checkpoint in the checkpoint control page. Later, the cleaner thread for the checkpoint file system catches the flag and scans the entire mapping for each block in the file system. If there is a mapping entry for the checkpoint, and if the previous checkpoint does not

have a mapping entry for the block, then the entry is moved to the entry for the previous checkpoint. Otherwise, if the mapping is not an allocation marker, the block is released. Then, all of the entries for the checkpoint after the one that is to be deleted are moved to the left.

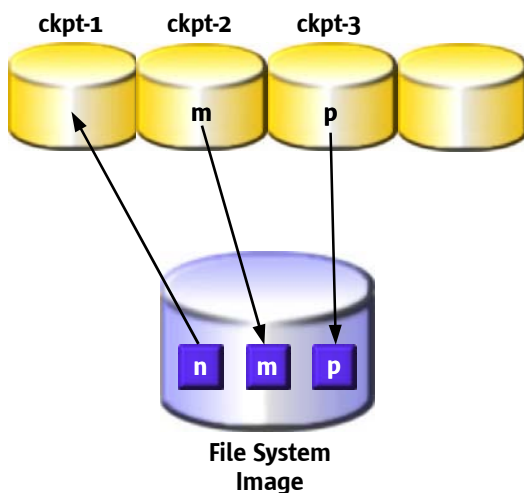
For example, in Figure 11, when deleting ckpt-2, *m* is moved to the entry for ckpt-1 and all of the entries after ckpt-2 are moved to the left (this operation is performed logically in the checkpoint stack, so the actual entries are not moved, only block ownership is delegated). Now, ckpt-3 replaces ckpt-2. After changing the mappings, the checkpoint stack is also updated similar to the method for checkpoint mapping entries.

Figure 11. Mapping for block *n* when deleting ckpt-1



Another example assumes ckpt-4 is to be removed. Because ckpt-3 contains no mappings for itself, *p* is copied to the entry for ckpt-3. This results in the mappings illustrated in Figure 12.

Figure 12. Mapping after deleting ckpt-4



Scheduling Checkpoints

Checkpoints can be created manually or automatically. Automatic checkpoints are created and removed based on the scheduling specified by the user. This scheduling is enforced by a checkpoint manager thread. However, the checkpoint manager does not control manually created checkpoints. Users can create manual checkpoints that are automatically removed by using the same naming convention the system uses for automatic checkpoints.

Accessing Checkpoints

Checkpoints are accessible in two ways — via the standard mechanism of mapping or mounting the virtual volume, or via the local directory checkpoint access feature. The local directory checkpoint access feature allows users to access any version of a directory by simply using normal client commands.

The checkpointed version of directories and files is accessed through a hidden, virtual directory named *.chkpnt*, within each live directory. Changing the current directory to the virtual *.chkpnt* directory enables users to access checkpointed or prior versions of file system objects. Note that *.chkpnt* directories are hidden to prevent problems with applications that search through hierarchies, such as backup or virus scanning applications. Users can access objects in the *.chkpnt* directories by explicitly navigating to them via a command line interface (CLI) or via a graphical user interface (GUI). For example:

```
# cd /live_directory/.chkpnt
```

Navigates the user to the checkpointed version of the directory *live_directory*, where the user can view prior versions of objects from *live_directory*. Users can also reference objects explicitly. For example:

```
# cp /live_directory/.chkpnt/ckpt-1/old_file1.txt /live_directory
```

Copies the *ckpt-1* version of *old_file1* back to *live_directory*. If a directory is removed and a user needs to access a checkpointed version of that directory, this can be accomplished by navigating first to the parent directory (of the removed directory) and then traversing the directory hierarchy until the desired version of the removed directory is reached. In the command sample below, the directory *d3* is removed but the user can access a checkpointed version of *d3* through the *.chkpnt* directory in its parent directory *d2*. In this example, *ckpt-1* is the name of the checkpoint that contains the desired version of the *d2* directory.

```
[/ ]# tree
--d1
  --d2
    --d3
3 directories, 0 files
[/ ]# cd d1/d2
[/d1/d2]# rm -rf d3
[/d1/d2]# cd ..
[/d1]# tree
--d1
  --d2
2 directories, 0 files
[/d1]# cd /d1/d2
[/d1/d2]# cd .chkpnt
[/d1/d2/.chkpnt]# ls
ckpt-1
[/d1/d2/.chkpnt]# cd ckpt-1
[/d1/d2/.chkpnt/ckpt-1]# ls
d3
[/d1/d2/.chkpnt/ckpt-1]# cd d3
[/d1/d2/.chkpnt/ckpt-1/d3]#
```

In order to access the *.chkpnt* directory, UNIX clients use standard file system commands or a GUI. The operation from a Windows client is similar. They can either provide a complete path name or use Windows Explorer. Due to the dissimilar methods employed by UNIX and Windows in accessing files, their respective clients access the checkpoints in different ways. UNIX clients simply NFS mount the checkpoint volume as with any other file system. Windows (CIFS) clients access the checkpoint via a share that is created by the administrator (shares are discussed in Chapter 5). To protect the files from unauthorized user access, file access permissions on the checkpoint are no different than they were when the checkpoint was created.

The Sun StorEdge 5000 family operating system is implemented with built-in safety measures to help ensure that data is available when checkpoints are active on a file volume. To prevent checkpoints from filling up the file system, the checkpoint manager does not create a checkpoint if the file volume is 90 percent full. In addition, when a checkpointed file volume is at 95 percent of its capacity, the checkpoint manager begins deleting checkpoints in chronological order until the volume is below 95 percent full, or until there is only one checkpoint remaining.

Compatibility Issues

In order to avoid name conflicts between user applications and existing files, each *.chkpnt* entry possesses a number of special characteristics. First, as previously discussed, it does not show up in directory listings. Applications not aware of the *.chkpnt* directories cannot see them, preventing potential problems with applications that traverse directory hierarchies, such as backup and virus scanning programs.

Second, if users already have existing files or directories named *.chkpnt*, these objects can retain their existing meaning by either renaming the objects or changing the access name for the checkpoint directories. The syntax to change the name of the *.chkpnt* directory is:

```
chkpnt vname {volname}
```

Shows the current hidden directory name for {volume}

```
chkpnt vname {volname} [new-name]
```

Sets the hidden directory name to *new-name*. Once the virtual checkpoint directory name is set with the *chkpnt vname* command, the file system does not allow any other objects to be created with that name.

Restoring Objects from Checkpoints

By definition, objects within a checkpointed volume can not be modified — this is the nature of checkpointing and desirable behavior. If a user needs to modify a checkpointed version of an object, an instance of it must first be returned to the live file system. Previously, the only way to accomplish this was to employ standard client-based copy mechanisms, such as drag-and-drop in a window or via the *cp* command in UNIX. Standard mechanisms are inefficient for a number of reasons:

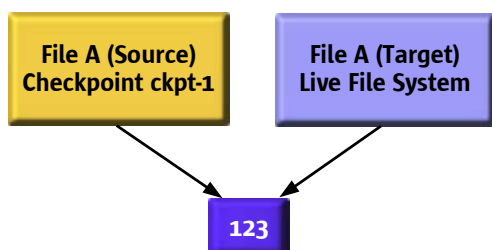
- The data is copied twice — from the file system to the client and from the client back to the file system.
- Network bandwidth is required.
- Client resources (CPU, network, memory) are used to execute the copy operation and therefore are not available for other processes.
- There is overhead at each layer of the operation.
- The copy operation is a block-for-block operation because the client system is unaware of the structure of the file system, and therefore is not aware of the underlying relationship between the blocks in the live and checkpointed versions of the file system.

The Sun StorEdge 5000 family operating system includes an internal *cp* command engineered to confine the operation to the file system, resulting in greater efficiency and speed.

The *cp* Command

The Sun StorEdge 5000 family command line interface (CLI) *cp* command is cognizant of the file system architecture and it can copy only those blocks that are not referenced in the live file systems using a copy-on-write implementation. For example, in Figure 13, after a *cp* restore operation, two versions of a file A — the LFS version and the checkpoint *ckpt-1* version — share the same block 123.

Figure 13. Sharing blocks between live and checkpoint file systems



In this case, when block 123 is referenced by either version of file A, the same block is accessed. If a client modifies block 123 on file A in the live file system, the checkpoint mechanism preserves the original block 123 in checkpointed versions of the file system. The syntax for the *cp* command is:

```
cp [-c] source destination
```

This command simply copies the source to destination. The source should be a regular file. If the destination is a regular file, the source overwrites it. Otherwise, if it is a directory, the source is copied to that directory. The *-c* option restores the checkpointed source file to the destination. If the destination is omitted, then the system tries to restore to the original (non-checkpointed) path. For example:

```
cp -c /v6.chkpt/ckpt-1/docs/sample.doc
```

restores to:

```
/v6/docs/sample.doc
```

The *cp* command is also available as part of the *chkpnt* command line operations. The syntax is identical except that the command is prefaced with *chkpnt*, and the *-c* option is not required:

```
chkpnt cp source [destination]
```

While a checkpoint file restore operation is in process, the file is locked for the duration of the operation and clients cannot perform any actions against it. Any attempt to access the file during a restore operation results in an error.

The CLI *cp* command is a general command and can be used to copy files on the Sun StorEdge 5000 family of NAS appliances for standard copy (to copy a file from one volume to another) or to restore a checkpointed version of an object to the live file system. If a checkpoint restore operation is interrupted for any reason — by a power failure for example — the operations resumes automatically when the system is restarted. Other checkpoint operations are also suspended during a checkpoint restore.

Chapter 4

Network Data Sharing

Sharing data between UNIX and Windows is a persistent challenge for network users and administrators. One large issue relates to the substantial differences between UNIX and Windows file access security models. Windows file servers use access control lists, or ACLs, that allow specific access rights for any number of users and groups. In contrast, NFS servers employ traditional UNIX permissions that provide access control for owner, group, and other. The goal of the Sun StorEdge 5000 family software is to provide an integrated security model in which a single file system can contain files with both Windows ACLs and UNIX file permissions.

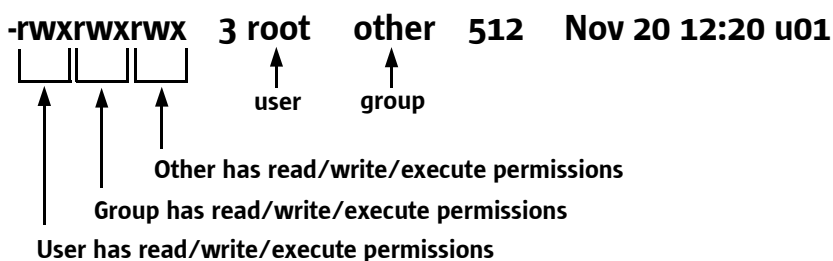
UNIX NFS Security Model

The NFS model is a distributed approach, where each individual system is responsible for maintaining security information. Each individual file and directory has an associated set of bits that specify:

- User ID (UID) of the owner
- Group ID (GID) of the owner
- Permissions bits for the owner, defining read, write, and execute (*rwX*) permissions
- Permission bits for the group, defining *rwX* permissions
- Permission bits for others, defining *rwX* for users other than UID or members of the group other than the group specified by the GID

Figure 14 illustrates the permissions on a UNIX file.

Figure 14. UNIX file permissions



To determine access levels, the NFS server first determines whether the request is from the file's owner, someone in the file's group, or other. It then uses the permission, group permission, or other permissions, respectively. Each UNIX server maintains its own database of users and groups defining UIDs and GIDs.

Windows CIFS Security Model

With the Windows Domain model, only a single server — the Primary Domain Controller (PDC) — is accountable for managing user information. With a centralized database of users and groups, Windows environments using the NT file system (NTFS) use a different file security model on the following information defined for each securable object (file or directory):

- Security ID (SID) of the owner
- SID of the owner's primary group
- Access Control List (ACL) for the file or directory

The ACL consists of a list of Access Control Entries (ACE) that define the read, write, and execute permissions also seen in UNIX, as well as change permissions (P), take ownership (O), delete (D), and others. With each ACE, the ACL defines a list of users and/or groups mapped to permissions.

Connection-Oriented vs. Connectionless

Another significant difference between the Windows (CIFS) and UNIX (NFS) paradigms is the notion of a connection or state between a client and server. The Windows approach can be generally classified as connection-oriented or stateful, where a connection is initially established and the client is in a known state of connectivity with the server. In order to access the network, a Windows Domain user authenticates a Primary Domain Controller (PDC) and establishes a connection. When accessing a remote server over the network, a Windows client establishes a session with the file server. In the event that the server is inaccessible, the client assumes that the connection or session is permanently broken. The client must then re-establish the session in order to access the server once again. The file locking functionality inherent to CIFS is another area where a clear state (locked or unlocked) is established between a client and the server.

Network communications between UNIX systems employing NFS are connectionless, in that they do not require a sustained connection between a client and the server. Each user gains access to the network by logging into a client. Unlike CIFS, where user authentication is performed centrally at any server, NFS user authentication is performed at each client. NFS clients establish a session between a client and the server with each file access and a temporary loss in connectivity does not necessarily break the link between them. The client simply attempts to re-establish the connection if it is broken. Therefore, with the connectionless approach utilized by NFS, clients tend to be less affected by server reboots.

File System Features for File Sharing

The Sun StorEdge 5000 family file system is designed to bridge between the two paradigms, with compatibility between the Windows and UNIX security models. The operating system supports both CIFS and NFS at the kernel-level, providing an integrated model for sharing data between UNIX and Windows systems, including network resolution via Network Information Services (NIS), Domain Name Service (DNS), Windows Internet Naming Service (WINS), ACL, and domain support.

In the Sun StorEdge 5000 family file system, files created via NFS are classified as UNIX files and files created via CIFS are classified as Windows files, maintaining a physical distinction between the two types of files. Windows files include security descriptors (ACLs), while UNIX files do not. For native file service requests (NFS requests to

UNIX files and CIFS requests to Windows files), the security model exactly matches the UNIX or Windows model. For non-native requests, the general rule for file access is that a user is not granted greater access via a non-native interface than is granted via the native interface.

Windows Security and Access Control Lists

The file system provides file access from Windows systems via CIFS with support for either share-level security or domain (ACL) security. In the share-level security mode, users are authenticated locally using a share password. When used in a domain environment, users are authenticated with a domain controller using a pass-through authentication scheme.

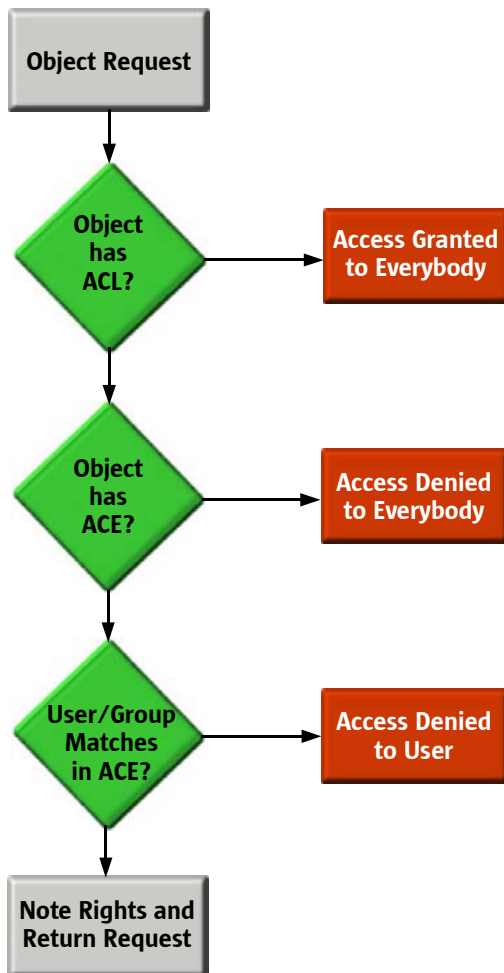
As with Windows systems, each NAS appliance defines a local domain on which it is a member. When users connect, they are authenticated using information obtained from the Primary Domain Controller (PDC). An access token identifies each user and contains the user's ID, the ID of each group of which the user is a member, the ID to be assigned to objects created by the user, and a default access control list. The access token represents users if they attempt to access a secured object (either files or directories).

Files are secured on the file system via security descriptors. Each security descriptor contains the ID of the file owner, the ID of the owning group, and an access control list. The ACL is a list of access control entries, each of which grants or denies access to a specific user or group.

CIFS Security Policy

The CIFS security policy implemented on the Sun StorEdge 5000 family of NAS appliances is the standard Windows security policy. Each request to access an object contains a set of desired access rights. These access rights are checked against the access control information defined in the file's security descriptor to determine whether access is granted or denied. The access validation method is depicted in Figure 15.

Figure 15. CIFS security policy



In addition to the Windows-like security policy, it is also important to note that the Sun StorEdge 5000 family of NAS appliances includes built-in Windows tools to manage file access rights (ACLs), providing tight integration with the Windows Domain security model.

Mapping a Windows ACL to UNIX Permissions

Whenever an ACL is created or modified, a set of complementary UNIX permissions is generated and stored in the file system to help ensure maximum file security for NFS file access. These generated permissions allow a reasonable level of NFS access without compromising the security policy of the CIFS ACLs. The following restrictions help maintain the proper security permissions:

- NFS users cannot change the permissions of a Windows file. For example, *chmod* operations are denied.
- NFS users cannot change the ownership of a Windows file. For example, *chown* operations are denied.

When a Windows file is created, the equivalent UNIX permissions listed in Table 1 are also generated. The permissions are mapped according to the following rules:

- If the security descriptor does not contain an ACL, the file is assigned UNIX permissions of *0777*, or “*rw-rw-rw-*”
- If the security descriptor contains an empty ACL, the file is assigned UNIX permissions of *0*, or “*-----*”
- If a group ACE is absent, the owning group is set to the user’s primary group and the group permissions are set to the same values as the other permissions

Table 1. Windows to UNIX permission bits mapping

	Windows	UNIX Equivalent
Directory	R	r
File	R	r
Directory	W	w
File	W	w
Directory	X	x
File	X	x
Directory	D	Ignored
File		
Directory	P	Denied
File		
Directory	O	Denied
File		

Mapping UNIX Permissions to Windows ACL

When a UNIX file is accessed via CIFS, an ACL is synthesized from the UNIX permissions in order to determine the appropriate access to grant. The restrictions for mapping UNIX permissions to ACL are as follows:

- If the owner UNIX ID does not map to a known user in the Windows domain, an owner SID is generated using the local UNIX domain SID. Otherwise, the appropriate owner SID is used in the ACL.
- If the owner UNIX GID does not map to a known group in the Windows domain, a group SID is generated using the local UNIX domain SID. Otherwise, the appropriate group SID is used in the ACL.

Permissions are mapped as described below and in Table 2.

- If the UNIX permissions are “*---*”, an *access denied* ACE is generated with its ACE rights set to “No Access”
- If the UNIX permissions are “*rw-*”, an *access allowed* ACE is generated with its ACE rights set to “Full Access”
- In all other cases, two entries are generated — an *access allowed* ACE to support permissions and an *access denied* ACE to prohibit other permissions

Table 2. UNIX to Windows permission bits mapping

UNIX	Windows Equivalent
r--	R
-w-	WD
--x	X
-wx	WXD

UNIX	Windows Equivalent
r-x	RX
rw-	RWD
rwX	Full
---	No Access

If the permissions of a UNIX file are modified via CIFS, e.g., an access control list is saved from a Windows system, the file becomes a CIFS file. An appropriate set of UNIX permissions is generated from the Windows discretionary access control list (DACL) and saved in the file or directory record. As with UNIX to Windows access, the general rule for multi-protocol file access applies — a user is not granted greater access via non-native interface (CIFS in this case) than is granted via the native interface (NFS).

DOS Attributes

In addition to supporting standard CIFS permissions, the Sun StorEdge 5000 family file system also supports DOS file attributes. The following DOS attributes are integrated with the file system and may be viewed, set, or cleared via Windows applications or utilities: *archive*, *hidden*, *read-only*, and *system*. The behavior associated with these attributes is:

- When a file is modified locally, via NFS or CIFS, the *archive* attribute is set. The archive attribute may be used for operations such as over-the-wire backup via Windows systems.
- If a UNIX file is created with a file name beginning with “.”, the *hidden* attribute is set. It is otherwise ignored by the file system.
- If the *read-only* attribute is set on a file, then both UNIX and Windows see it as *read-only*, regardless of the UNIX or Windows permissions.
- The value of the *system* attribute is ignored by the file system.

Cross-Platform File Locking

True cross-platform file access is not complete or secure without cross-platform file locking mechanisms. The Sun StorEdge 5000 family file system employs a kernel-integrated file locking mechanism that supports the following file locking schemes:

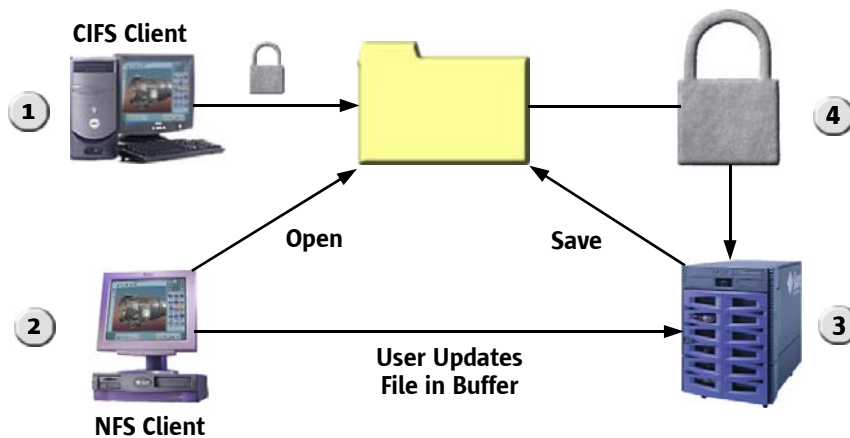
- UNIX byte-range locks (advisory locks)
- Windows (CIFS) whole-file locks
- Windows (PC)NFS whole-file locks
- Windows (CIFS) byte-range locks
- Windows (PC)NFS byte-range locks
- Windows (CIFS) Opportunistic locks (optional file locks acquired at file open time, breakable by CIFS file access)
- Cross-protocol file locking

In addition to supporting single-protocol file locking, the file system is also designed to protect user data with the support of a kernel-integrated, cross-platform file locking mechanism. This functionality helps ensure that concurrent data access from first CIFS and then NFS preserves the integrity of the file.

Figure 16 demonstrates how the file system enforces locks between CIFS and NFS with the following process:

1. The CIFS client opens a file on a Sun StorEdge 5000 family file system, initiating a file lock
2. A UNIX client opens the same file
3. The file is updated in the UNIX client system's buffer and the user attempts to save the changes
4. The file system denies the write operation due to its locked state

Figure 16. Enforcing locks between CIFS and NFS



File Locking Caveats

File locking mechanisms in the Sun StorEdge 5000 family file system only *permit* file locking. The responsibility of actually applying the lock is entirely up to the application that is used to access the file. In other words, file locking is engaged by the application rather than set explicitly by the storage system.

Managing Space

On any shared file system, it is important to set limits on how much disk space individual users can consume to keep one user or group of users from using all of the available disk space. The Sun StorEdge 5000 family of NAS appliances enable the administrator to set user and group quotas to determine how much disk space is available to a user or a group and how many files they can write to a volume. It is also possible to set directory tree quotas to regulate how much space is available for a specific directory and how many files it can contain.

Unicode Support

The Sun StorEdge 5000 family of NAS appliances supports Unicode, the double-byte encoding system that theoretically allows encoding of all characters in all languages of the world. Unicode is supported for directory names, file names, and file content. Unicode is also supported for user names and passwords for both Windows clients.

Chapter 5

Shares and Support for ADS and Dynamic DNS

To support Windows 2000 and above, the Sun StorEdge 5000 family file system provides the ability to publish shares in Active Directory Services (ADS) and Dynamic DNS. There are two types of shares — static shares and autohome shares. Static shares are persistent shares that remain defined regardless of whether or not users are attached to the server. Autohome shares are temporary shares that are created when a user logs on to the system and removed when the user logs off.

Shares

A shared resource, or share, refers to a local resource on a Windows server that is accessible to clients on the network. On a NAS server, it is typically a file volume or a sub-tree within a volume. Each share is assigned a name by which it is referenced on the network. To clients on the network, the share appears as a complete volume on the server, and they do not see the local directory path directly above the root of the share.

A common use of shares is to provide network access to home directories on a network file server, where each user is assigned a home directory within a file volume. Shares are then created to enable users to assign their individual home directories as network drives on their workstations. For example, a volume *vol1* may contain a home directory named *home* and subdirectories for users Tom and Mary. For this example, the shares are defined in Table 3.

Table 3. Home directory shares

Share Name	Directory Path
tom	/vol1/home/tom
mary	/vol1/home/mary

Note – Windows Workgroup uses share-level security administered UID, GID, and passwords. With Windows Workgroup, the rights to a directory are determined by the share definition rather than by the user. The file system assumes the client performs no authentication and explicitly asks for permission through the use of a password with every share-connection request.

With a large number of users, maintaining home directory shares can be a time-consuming administrative task that must be performed whenever user accounts are added or removed. In network environment with thousands of users, thousands of shares must be created and administered.

Autohome Shares

The CIFS autohome share feature eliminates the administrative task of defining and maintaining home directory shares for each Windows user who accesses the system. The system creates autohome shares when a user logs on and removes them when the user logs off. When browsing shares via Network-Neighborhood, only statically defined shares and autohome shares for *connected* users are listed. Using the autohome feature, rather than maintaining a statically defined share for each user account on the PDC, reduces the administrative effort required to maintain user accounts and increases the efficiency of server resources by reducing the list the browser needs to display.

The autohome path is the base directory path for the directory shares. For example, if a user's home directory is */vol1/home/tom*, the autohome path is */vol1/home* and the temporary share is named *tom*. The user's home directory name must be the same as the user's logon name.

When a user logs on, the server checks for a subdirectory that matches the user's name. If it finds a match, and that share does not already exist, it adds a temporary share. When the user logs off, the server removes the share. Windows clients may automatically log a user off after fifteen minutes of inactivity, which results in the autohome share disappearing from the list of published shares. This is normal CIFS protocol behavior. If the user clicks on the server name or otherwise attempts to access the file system, the share automatically reappears.

Note – All autohome shares are removed when the system reboots.

Support for Active Directory Service (ADS) and Dynamic DNS

To support Windows 2000 and above, the Sun StorEdge 5000 family file system provides the ability to publish shares in Active Directory Services (ADS) and Dynamic DNS. Active Directory is the Windows 2000 directory service that provides access to domain information such as users, groups, and shared resources. Active Directory clients access this information on the network using the Lightweight Directory Access Protocol (LDAP). The ADS relies on the Internet Domain Name System (DNS) to provide name resolution services. DNS is the industry standard service used throughout the Internet to resolve host names to Internet Protocol (IP) addresses. To locate an Active Directory server, an Active Directory client queries DNS. For example, to log on to an Active Directory domain, an Active Directory client queries its configured DNS server for the IP address of the LDAP service running on a domain controller for a specified domain. The DNS provided with ADS supports the ability for clients to dynamically update their entries in the DNS database, which is known as dynamic DNS.

After shares are published in ADS, ADS clients can access them by looking in the Active Directory. Shares appear in ADS as folder objects that point directly to the appropriate shares on the Sun StorEdge 5000 family file system. Shares can be put into any container in which a share folder object can be created. One of the properties of a share folder object is its Universal Naming Convention (UNC) name, which specifies both the server on which the share exists and the name of the share: *\\company.name.com\sharename*. ADS offers the ability to control access to share objects within the directory service, in addition to any access controls enforced on the shared directory.

The Sun StorEdge 5000 family of NAS appliances supports Kerberos version 5 authentication for secure ADS and dynamic DNS updates. Kerberos uses secret-key cryptography to provide a network authentication protocol for client/server applications. Each secure ADS or dynamic DNS update is performed within the context of an ADS user.

The system communicates with a Key Distribution Center (KDC), which normally resides on a domain controller, to authenticate the user prior to performing an update.

Configuring ADS

ADS must be enabled to publish or remove shares in the ADS. The ADS configuration options available with the Sun StorEdge 5000 family of NAS appliances are listed in Table 4.

Table 4. ADS configuration options

Option	Description
Enable	Enable/disable ADS. ADS must be enabled to publish or remove shares in the ADS.
ADS Domain	The Windows domain to which the NAS appliance belongs.
User Name	A Windows user who has the administrative rights required to perform ADS updates.
Password	Password for the ADS administrator user.
User Container	The ADS location of the user container (in LDAP distinguished name notation — without the domain).
Kerberos Realm	The name of the Kerberos realm for secure ADS or secure dynamic DNS update. This is generally the ADS domain or DNS domain in uppercase, e.g., COMPANY.COM
KDC Server	The host name of the Kerberos Key Distribution Center (KDC) server, usually a domain controller. This field can be left blank if clients can find the KDC via DNS.

Additional information on the options in Table 4 are provided below:

- The ADS domain is equivalent to concatenating the Windows domain name and the DNS domain name. For example, if the Windows domain is *SALES* and the DNS domain name is *company.com*, the ADS domain name is *sales.company.com*.
- The user name and password provide the user credentials used to authenticate ADS updates. The user must be a valid Windows user, defined in ADS, who has the administrative rights required to perform secure ADS updates, such as administrator.
- The *User Container* field specifies the container in which the user name resides. For example, if the *administrator* user resides in the *users* folder, then the field format is an LDAP distinguished name notation without the domain. Thus, for the administrator user the container is specified as: *cn=users*.
- If the user definition resides in an organizational unit folder (for example, *user_folder*), then the container is specified as: *ou=user_folder*.
- If the user definition resides in a sub-container folder of a parent folder, then the container is specified as: *ou=sub_folder,ou=parent_folder*. Note the reverse order definition.
- Containers that are not organizational units must use the “*cn=*” naming attribute.

Configuring Dynamic DNS

The Sun StorEdge 5000 family of NAS appliances provides support for dynamic DNS, which can be used to automatically update the IP address and host name of a Sun StorEdge NAS appliance. In an ADS environment, it is impor-

tant that the DNS information is correct since ADS clients use DNS to locate the NAS appliance when they access shared resources.

The system updates DNS when it boots up or when DNS settings are saved. Dynamic DNS must also be enabled in DNS. The system can update a DNS using secure or non-secure mechanisms. It first attempts to update DNS using the non-secure method. If this fails, it tries the secure method. In order for the system to update a DNS zone securely, the zone must be an ADS integrated zone, meaning it is stored in ADS. The system uses Kerberos to authenticate an ADS user before DNS can be updated securely. The dynamic DNS configuration options are listed in Table 5.

Table 5. Dynamic DNS configuration options

Option	Description
Enable	Enables or disables the dynamic DNS updates
User Name	The name of a Windows 2000 user that has the appropriate rights to perform secure dynamic DNS updates, e.g., the administrator
Password	Password for the specified user name

In order for secure DNS to work, the Kerberos Realm must be configured in ADS and the DNS zone that is to be updated must be stored in ADS.

Publishing SMB/CIFS Shares in ADS

If ADS is enabled, shares defined via the SMB/CIFS Share Setup screen can be published in ADS. The SMB/CIFS Share Setup screen contains a field to specify the ADS location in which to publish each share. This field is specified using standard LDAP distinguished name notation — without the domain name. For example, to publish the share in the organization unit named *folder*, the container is specified as *ou=folder*. When the share definition is saved, it is published in the specified ADS container. This field is optional — if it is left blank, the share is not published in ADS.

Additional containers, called organization units, can be created in ADS and can be used to hold share folder objects. The following are examples of the format used for ADS containers:

- *ou=share_folder*
- *ou=company_share, ou=share_folder*
- *cn=computers*

A share is removed from ADS when it is deleted on the system or when the ADS container field is cleared of the share definition. A previously unpublished NAS appliance share can be published by entering the ADS container data on the setup screen. ADS is updated when the share information is saved — assuming that ADS is enabled on the system.

The Sun StorEdge 5000 family of NAS appliances uses secure ADS updates to publish and remove shares, which requires that the ADS user, specified in the ADS setup screen, be authenticated via Kerberos v5. The user must exist in ADS and must have administrative rights to create or delete a share folder object. DNS is queried to locate a domain controller on which to perform updates. If a previously located domain controller becomes unavailable, the system looks for another domain controller within its primary domain.

Chapter 6

Conclusion

Unlike traditional file servers based on Windows or Linux operating systems, the Sun StorEdge 5000 family of NAS appliances is designed to provide kernel-integrated support for both UNIX and Windows files, allowing a single platform for file sharing. Rather than managing multiple file servers on different platforms, IT departments can now consolidate corporate data onto a single, high-availability platform — simplifying data management and possibly leading to lower TCO. The unique and innovative elements of the Sun StorEdge 5000 family software help reduce costs, streamline administration, and enhance productivity:

- **Journaling file system** — Helps ensure data consistency and fast reboots.
- **On-line expansion** — Enables the ability to add capacity without compromising data availability.
- **Checkpoints** — Enables instantaneous online backup and fast restores while keeping primary data available to users. Ideal for 24 x 7 operations.
- **Full multi-platform support for UNIX and Windows** — Compatibility with Windows Domains, ACLs, and UNIX NIS.
- **Multi-protocol data sharing and file locking** — Enables sharing and consolidation of files on one platform.
- **Unicode support** — Enables deployment throughout the world.

With multi-protocol support for file sharing, scalability, reliability, file locking, easy backup and restores, and easy manageability — the Sun StorEdge 5000 family of NAS appliances is the ideal file storage platform for today's demanding users and applications.

Chapter 7

References

For more information on topics discussed in this paper, please refer to the following sources.

Sun Web Sites

- <http://www.sun.com/storage/5000>

From <http://docs.sun.com>:

- *Sun StorEdge™ 5210 NAS Software Installation, Configuration, and User Guide*
- *Sun StorEdge™ 5310 NAS Software Installation, Configuration, and User Guide*

Chapter 8

Glossary

ACE

Each Active Directory object is protected by access control entries (ACEs). ACEs identify which users or groups can access objects. Each ACE contains the SID of each user or group that has permission to access that object and defines what level of access is allowed. For example, a user might have read-only access to certain files, read-and-write access to others, and no access to others.

ADS

Active Directory Service (ADS) is a Windows name service integrated with Domain Name Service (DNS). ADS runs only on domain controllers. In addition to storing and making data available, ADS protects network objects from unauthorized access and replicates objects across a network so that data is not lost if one domain controller fails.

DAACL

Discretionary Access Control List (DAACL). An access control list that is controlled by the owner of an object to specify the access particular users or groups can have to the object.

Domain Controller

A computer running Windows that stores domain-wide directory data (such as system security policies and user authentication data) and manages user-domain interactions, including user logon processes, authentication, and directory searches.

DNS

Domain Name Service (DNS) resolves domain names to IP addresses. This service allows the system to be identified by either its IP address or its name.

Dynamic DNS

A method of keeping a domain name linked to a changing IP address as not all computers use static IP addresses.

Inode

A data structure holding information about files in a UNIX file system. Each file is uniquely identified by the file system in which it resides and its inode number on that file system.

KDC

Key Distribution Center (KDC). In Kerberos authentication, the KDC maintains a list of user principals and is contacted through the *kinit* program for the user's initial ticket. Frequently, the KDC and the Ticket Granting Service are combined into the same entity and are simply referred to as the KDC. The Ticket Granting Service maintains a list of service principals and is contacted when a user wants to authenticate to a server providing such a service. The KDC is a trusted third party that must run on a secure host. It creates ticket-granting tickets and service tickets.

Kerberos

A network authentication service developed under Massachusetts Institute of Technology's Project Athena that strengthens security in distributed environments. Kerberos is a trusted third-party authentication system that relies on shared secrets and

assumes the third party is secure. It provides single sign-on capabilities and database link authentication (MIT Kerberos only) for users, provides centralized password storage, and enhances PC security.

LDAP

Lightweight Directory Access Protocol (LDAP). An Internet Engineering Task Force (IETF) protocol for accessing on-line directory services.

Metadata

Data about data.

NFS

Network File System (NFS) is a client/server application that lets a computer user view and optionally store and update files on a remote computer as though they were on the user's own computer.

NIS

Network Information Service (NIS) creates a central database on the Solaris™ Operating System for host, user, and group information. It maintains this database and administers access to resources based on the user's group and host information.

NIS+

Network Information Service Plus (NIS+) designed to replace NIS, and is the new default naming service for the Solaris Operating System. NIS+ can provide limited support to NIS clients, but is mainly designed to address problems that NIS cannot address. Primarily, NIS+ adds credentials and secured access to the NIS functionality.

NTFS

NT File System, one of the file systems used by the Windows operating system (Windows also supports the FAT file system).

Organization Units (OU)

In Windows 2000 and above operating systems, organizational units (OUs) are a type of directory object into which users, groups, computers, printers, shared folders, and other organizational units within a single domain can be placed. An organizational unit enables the administrator to logically organize and store objects in the domain. If there are multiple domains, each domain can implement its own organizational unit hierarchy.

PDC

Primary Domain Controller (PDC) is a Microsoft Windows server responsible for handling all accounts in a domain.

SID

A unique number created by the security subsystem of the Windows 2000 and above operating systems, and assigned to security principal objects, such as user, group, and computer accounts. Internal processes within the Windows operating system refer to an account's SID rather than to the account's user or group name.

UNC

Used in IBM PC networking to completely specify a directory on a file server. The basic format is: \\servername\sharename.

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA Phone 1-650-960-1300 or 1-800-555-9SUN Web sun.com



Sun Worldwide Sales Offices: Argentina +5411-4317-5600, Australia +61-2-9844-5000, Austria +43-1-60563-0, Belgium +32-2-704-8000, Brazil +55-11-5187-2100, Canada +905-477-6745, Chile +56-2-3724500, Colombia +571-629-2323, Commonwealth of Independent States +7-502-935-8411, Czech Republic +420-2-3300-9311, Denmark +45 4556 5000, Egypt +202-570-9442, Estonia +372-6-308-900, Finland +358-9-525-561, France +33-134-03-00-00, Germany +49-89-46008-0, Greece +30-1-618-8111, Hungary +36-1-489-8900, Iceland +354-563-3010, India-Bangalore +91-80-2298989/2295454; New Delhi +91-11-6106000; Mumbai +91-22-697-8111, Ireland +353-1-8055-666, Israel +972-9-9710500, Italy +39-02-641511, Japan +81-3-5717-5000, Kazakhstan +7-3272-466774, Korea +82-2-2193-5114, Latvia +371-750-3700, Lithuania +370-729-8468, Luxembourg +352-49 11 33 1, Malaysia +603-21161888, Mexico +52-5-258-6100, The Netherlands +00-31-33-45-15-000, New Zealand-Auckland +64-9-976-6800; Wellington +64-4-462-0780, Norway +47 23 36 96 00, People's Republic of China-Beijing +86-10-6803-5588; Chengdu +86-28-619-9333, Guangzhou +86-20-8755-5900; Shanghai +86-21-6466-1228; Hong Kong +852-2202-6688, Poland +48-22-8747800, Portugal +351-21-4134000, Russia +7-502-935-8411, Saudi Arabia +9661 273 4567, Singapore +65-6438-1888, Slovak Republic +421-2-4342-94-85, South Africa +27 11 256-6300, Spain +34-91-596-9900, Sweden +46-8-631-10-00, Switzerland-German 41-1-908-90-00; French 41-22-999-0444, Taiwan +886-2-8732-9933, Thailand +662-344-6888, Turkey +90-212-335-22-00, United Arab Emirates +9714-3366333, United Kingdom +44-1-276-20444, United States +1-800-555-9SUN or +1-650-960-1300, Venezuela +58-2-905-3800, or online at sun.com/store

SUN™ THE NETWORK IS THE COMPUTER © 2004 Sun Microsystems, Inc. All rights reserved. Sun, Sun Microsystems, the Sun logo, Solaris, Sun StorEdge, and The Network Is The Computer are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd. Other brand and product names are trademarks of their respective companies. Information subject to change without notice. Printed in USA 12/04 XX0000-0/#K