Sun™ Cluster 3.0 Software Cluster File System (CFS): Making the Most of the Global File Service

Technical White Paper

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
1 (800) 786.7638
1.512.434.1511

Please
Recycle

Table of Contents

CHAPTER 1

Introduction

Application downtime or poor performance are no longer acceptable in a world of 24x7 access, e-commerce, the Internet, and application service providers. When customers can switch suppliers at the click of a button, high levels of service are vital. Sun™ Cluster 3.0 software is designed to help businesses meet the dual challenges of maintaining high levels of application availability and scalability while avoiding the increased overhead often encountered when deploying multinode clustered systems.

Sun Cluster 3.0 software differs substantially from previous versions in three distinct respects. First, global devices allow disks, tapes, CD-ROMs, and meta-devices (which are created by volume management products) to be attached to any node in the cluster, and accessed transparently from every node as if the devices were local. Second, global networking provides users with a single address they can use to connect to a highly available system service. Finally, global file service allows UFS or HSFS file systems to be mounted concurrently and coherently on every cluster node, and accessed as if the systems were mounted locally. These features are developed with the Solaris™ Operating Environment cluster extensions.

This white paper explains how the global file service (a term used interchangeably with the cluster file system, CFS) can be used in the implementation, configuration, ongoing system management, and capacity planning of a Sun Cluster 3.0 system. A high-level view of these components does not list specific commands or procedures, which are covered in the Sun Cluster 3.0 documentation set. Finally, the paper describes how applications can be written to take best advantage of the product's new features.

The guidelines given in this document apply only to Sun Cluster 3.0 software, update 1 clusters running onthe Solaris 8 Operating Environment, update 4 (04/01) or later.

CHAPTER 2

Global Devices: Homogeneous Access to Single- and Dual-ported Devices

A clustered system has a number of I/O devices connected to its constituent nodes. These can be singly attached (local to a node), or dual-hosted between two nodes[1]. It is important to understand that a device need not be dual-hosted to be considered a global device. It simply means that, in the case of a device that is connected to only one node, the failure of that node will render the device unavailable. If, however, the device is dual-hosted, the underlying cluster framework will transparently switchover the active I/O path to the remaining connection.

This notion of active (primary) and inactive (secondary) I/O paths is used throughout this document. I/O operations that are performed through these paths are made highly available via a checkpointing/minitransaction framework. This ensures that operations failing to complete — due to events such as node failures — are transparently restarted. The mechanism is required only for operations that modify system state in some way. For example, creating a file will necessitate its use because, if the primary node failedl arbitrarily during the operation, the secondary node would need to know if: a) the file existed before the primary node failed, and b) the primary node successfully completed the creation. Either way, it must return a file handle or a UNIX® error to the calling process. In cases that do not modify system state, such as reading data pages or file attributes, no such overhead is necessary, because the calling node can simply reexecute the desired method when the initial call fails.

Singly attached devices have no overhead, because there is no alternative path through which the I/O could be completed. Instead, a failed request simply returns a UNIX error directly to the application. The same is true when the final path is lost to a dual-hosted device.

1. Sun Cluster 3.0 is persistence group reservation (PGR-3) ready. Availability of PGR-3 devices will enable greater than two-host configurations.

CHAPTER 3

The Global File Service: An Introduction to CFS

This section describes the inner works of the cluster file system (CFS), which is important to understand when deploying Sun Cluster 3.0 software. Familiarity with the salient points of the implementation provides a context and explanation for the recommendations made here.

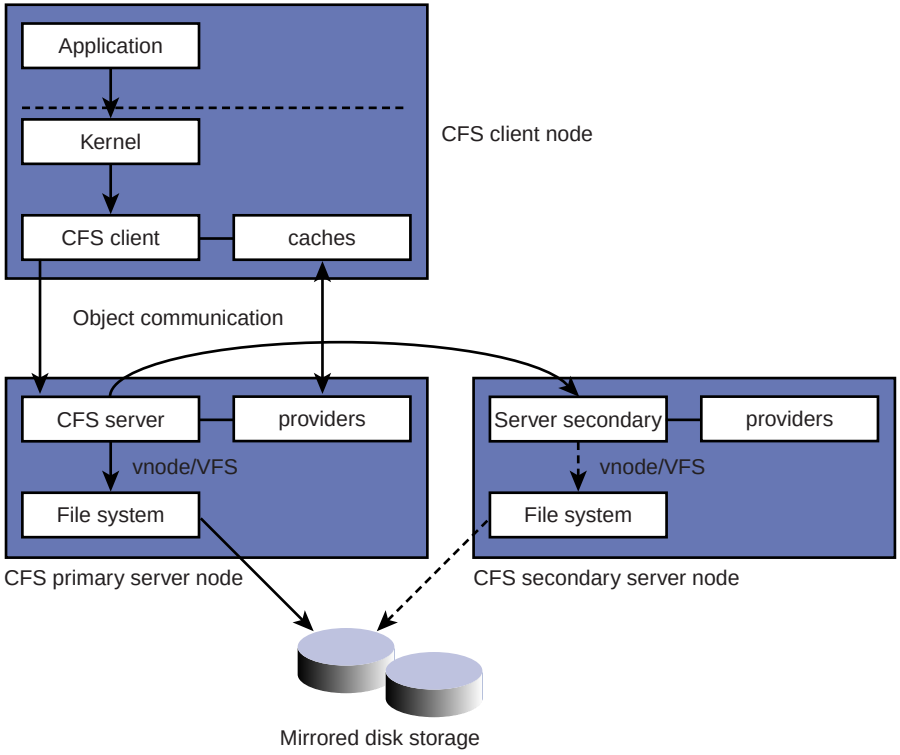

FIGURE 1 CFS Architecture

CFS is a highly available, distributed, cache-coherent file system that allows UFS or HSFS file systems to be concurrentlyaccessed on multiple cluster nodes. CFS has a kernel-based, client/server architecture (See FIGURE 1) built on the vfs/vnode interface to minimize the changes needed to either UFS, HSFS, or file systems that may be supported in the future[2].

User applications communicate with UFS or the CFS client object through the vnode/vfs interface using standard system calls such as open, close, read, write, and so on. The kernel is responsible for linking the appropriate type of vnode structure into its internal file system object representation when the initial mount operation takes place. When a normal UFS file system is mounted, then files within it have standard UFS vnodes on which I/O operations act[3]. Alternatively, if a file system is mounted globally, either using the `mount -g` command, or the global flag in the `/etc/vfstab` entry, then the files within it will have CFS client vnodes on which the I/O operations are performed. Operations on the latter will initially invoke communication with the CFS server objects, instead of UFS. On the node hosting the primary I/O path, the local CFS server object will then call the appropriate UFS vnode operations to satisfy the original request before sending the data or file attributes back.

2. For example, Veritas File System (VxFS).

3. Mounting a global device as a local file system is not supported, as it hinders switchovers. With this kind of mount, switchovers of the device service will fail, because the device will be in use on the primary node.

It is the CFS server's responsibility to perform the necessary physical I/O operations and drive coherency management of the Solaris pages and file attribute caches in the CFS client layer. This coherent cache management ensures that I/O operations are performed only when absolutely necessary to improve system performance.

The read-modify-invalidate model used by CFS is similar to that found in the CPU caches of an symmetric multiprocessing (SMP) server. To allow cached copies of file system data pages and attributes to be held on multiple nodes concurrently, three additional kernel structures are needed. These structures manage the tags required to track the state of the particular items. This overhead is relevant to anyone performing capacity planning exercises and is covered in the section, "Calculating Memory Overheads."

Read and Write Operations Under the Covers

When data is not cached by the local CFS client, a physical I/O operation will be performed to retrieve it from disk. Here, messages pass between the CFS client and server objects on their respective nodes using TCP-IP. When the data is on a locally hosted global file service, or cached locally, these messages will be optimized to direct kernel function calls.

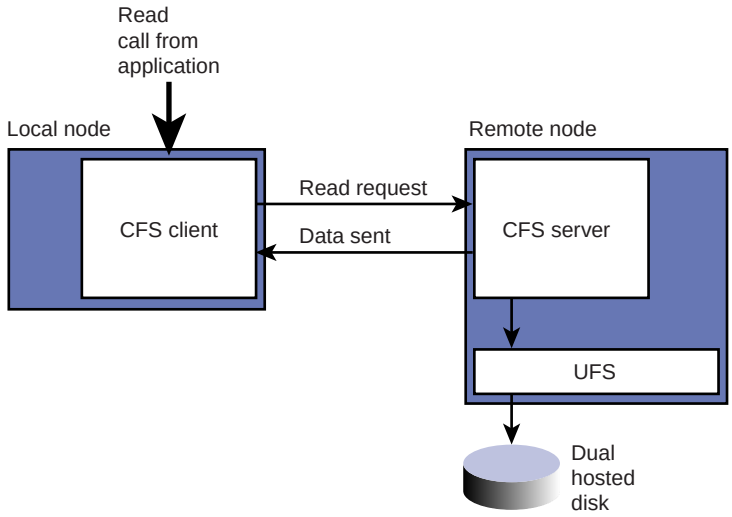

FIGURE 2 IDL Read Requests

The overhead of acquiring data from a remote node means that the performance of remote reads is inevitably lower than for local reads (when the data is not cached). However, if sufficient private interconnect bandwidth is available, multiple transfers can be handled in parallel, ensuring the scalability of the overall read throughput.

Remote writes are implemented by the server reading data from the client. This ensures that kernel buffer space is available on the server to which the data is being transferred. It also adds a TCP-IP message to the overall cost of a write. This slows down the performance of writes, but given sufficient private interconnect bandwidth, the write throughput should scale with the number of parallel requests.

Individual read or write requests, depending on their size, may be fragmented by the kernel drivers into smaller requests and then sent serially over a single private interconnect. However, separate I/O requests are distributed on a round-robin basis between all the available private interconnects. So, although adding interconnects will not accelerate an individual I/O operation, they will improve overall throughput. Exploiting this parallelism is a key factor in getting the most out of the global file service.

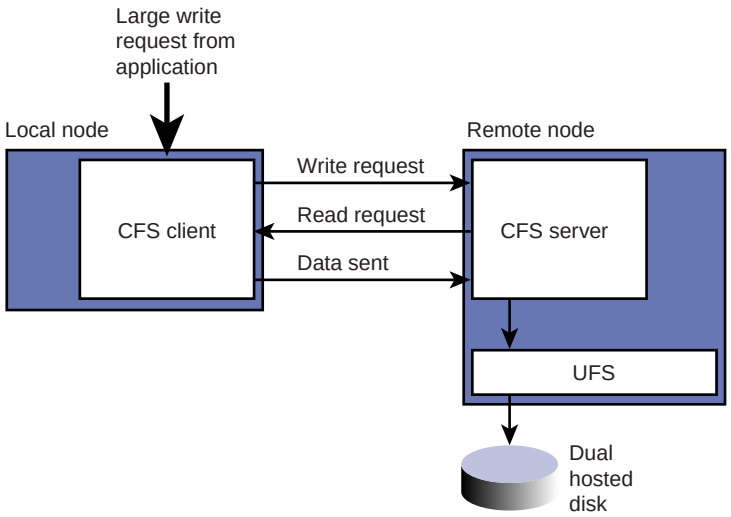

FIGURE 3 CFS Mechanism for Large Write Requests

The general recommendation is that HA applications should use locally hosted global file services whenever possible. Scalable services will perform remote reads and writes. Unless these are particularly high in volume, they should not pose a performance problem. This will apply until next-generation, high-performance, Remote Shared Memory (RSM) transports are available.

The File Attribute and Page Caching/Flushing Mechanism

Caching of both file system data pages and file attributes is critical to maintaining overall cluster performance. Internally, Sun Cluster 3.0 software uses a system of tokens to manage the possession and invalidation of the caches concurrently accessed from multiple cluster nodes. When a file system is mounted globally, pairs of cached objects on the client and cache provider objects on the server are linked together. Among these are a pair of file attributes and a pair of data pages for each active file in the CFS.

When data pages are read from a file or its attributes are being checked, the CFS client will first need to acquire a read token for that file from the relevant CFS server. Once the token has been acquired, then the data can be read in. If subsequent references are made to either that same data orits attribute and the token is still valid, then the data can be read out of the local caches, rather than from the CFS server. Multiple nodes can concurrently hold read tokens for data pages from the same file or attributes of the same file.

When a client on a node changes the attributes of a file or writes to it, the client mustfirst acquire the corresponding write token. Unlike the read token, only one node can hold the write token for the data or attributes of each file. Before this token is granted, the server invalidates all currently held read tokens and cached data associated with them. Subsequent requests for these invalidated pages or attributes will necessitate that data be reread from the CFS server.

While there is little that can be done from a configuration standpoint to take advantage of this situation or avoid its ramifications, it should be recognized that read-oriented workloads are likely to scale better than those performing a significant number of writes to shared files.

File System Mount Option

Global

The global flag allows a file system (UFS or HSFS) to be mounted concurrently on all nodes currently participating in the cluster.

Logging

Solaris 7 software introduced a logging UFS file system. The logging option is needed to ensure rapid file system recovery after a fail-over[4] of a UFS file system. Even if this mount option is not used, it is inserted automatically by the CFS mount if a UFS file system is mounted globally.

Logging is also recommended for all other local file systems to speed up recovery after a system crash.

Syncdir

When a UFS file system is mounted globally it can be done with or without the syncdir[5] option. The decision to use it or not will depend on whether absolute performance or error semantics take precedence.

The syncdir option comes into effect only when a file is being extended, either by data being appended to the end, or by a previously unfilled page being allocated (a hole). With the syncdir option set, notification of the operation is flushed directly to disk and not held in the in-memory logs. The write operation will not return until the data allocation is physically committed to disk. If the file system runs out of space during this operation, then an error will be returned to the write(2) call, even if the failover of the primary I/O path happens mid-call. The downside of this procedure is that the performance of the write(2) call is substantially degraded. For remote I/O operations, this can be as much as a factor of 10.

Without the syncdir option, CFS will reserve space for the write(2) operation, but the transaction will be held in memory and not committed to disk immediately. However, the write(2) call will return and, if there is sufficient space, do so without raising an error. If there is a fail-over before the in-memory log is flushed to disk, and the file system fills up on the new primary before the dirty page is flushed out by the CFS client, then when CFS tries to reserve space later on, it will find there is none available. Under these circumstances, CFS will return an ENOSPC error from the close(2) call. In cases where the application flushes data using the sync(2) call, ENOSPC would be returned from sync(2).

This behaviour without the syncdir flag is similar to that of NFS, but is less likely to occur because of the small time window in which the two events would have to take place. The syncdir mount option should be used only if the application cannot handle ENOSPC errors from sync(2) or close(2), and if the failure of the application to handle such errors is critical.

4. Fail-over is the term used to describe the movement of services and disk sets after a node failure. Switch-over is where this movement is performed under administrative control. The use of fail-over through this document will imply switch-over as well.

Forcedirectio

The forcedirectio[6] option, as the name suggests, forces I/O operations to be transferred directly from the user address space to disk, bypassing the kernel page cache. The option is particularly beneficial to applications that do their own caching, such as a relational database systems (RDBMS). Note that the forcedirectio option applies to all files opened on the specified file system, and may have a negative impact on other applications using the file system that do not cache data themselves.

Recommended Mount Flag Combinations for Global UFS Mounts

Table 1 provides recommendations on which combinations of mount options to use for various classes of application.

[Table 1] Recommended flags for various classes of application.

Application Class	Mount Option Flags
NFS Data	global, logging
Web Server Data	global, logging
RDBMS Data, Logs	global, logging, forcedirectio
Applications Binaries, Configuration Files	global, logging

5. This option applies only to global UFS mounts.

6. This option applies only to global UFS mounts, and is applicable to standard Solaris software.

CHAPTER 4

Considerations for System Setup and Configuration

General Tips for Maximizing I/O Performance

This section provides tips for configuring disk volumes, creating file systems, and choosing a private interconnect infrastructure to maximize I/O performance.

Configuring disk volumes

As with a single server, the choice of disk and configuration of disk subsystem will impact the I/O performance of the system as a whole. Sun Cluster 3.0 software does not change the standards used when configuring these systems. It is still important to stripe disks with a suitable interlace (usually 32 Kbytes), mirror them for resilience, and add Dirty Region Logs (DRLs) to Veritas Volume Manager (VxVM) disk groups to speed the resynchronisation of the mirror components after a fail-over. Sun Cluster 3.0 does not add performance considerations over and above those that normally apply.

In general, I/O performance characteristics of the system are independent of which volume management product is employed: Sun Volume Manager or Veritas Volume Manager. Because it does not perform a device discovery operation on the new primary before starting the volumes, Sun Volume Manager offers slightly faster volume fail-over times.

Finally, a logging file system is available through two mechanisms. First, a UFS file system can be mounted on a Sun Volume Manager trans-device. Second, the standard Solaris 8 logging file system option can be used. The latter is the preferred option.

Creating File Systems

While creating file systems and installing software are not frequently performed on a production cluster, it is important to understand how to complete them as quickly and efficiently as possible. The points raised below are relevant to applying software patches or growing a file system (see `growfs(1M)`) which are more common administrative requirements.

Whether creating or growing a file system, installing or patching software I/O is performed on the underlying disk subsystem. When using Sun Cluster 3.0 software, with its global file service capability, it is possible that the devices to which data is being written are remote. If they are, then a performance penalty will manifest itself as slower writes, increased CPU and interconnect utilization. So whenever these operations are performed, it should be from the node that hosts the primary path to disk devices. Preferably, this is achieved by logging directly onto the node that hosts the primary path and executing the command there. Alternatively,it can be achieved by switching the diskset's[7] primary path (using `scswitch -D`) to the node from which the commands are to be run.

There is no additional overhead or benefit attached to any of the `newfs(1M)` options when used with CFS, ebyond those encountered on a nonclustered system.

Choosing the Private Interconnect Infrastructure

Sun Cluster 3.0 software supports either 100BaseT Ethernet or Gigabit Ethernet (1000BaseSX) as private interconnect infrastructures.

The primary functions of the private interconnect are to provide a transport mechanism for:

- Cluster Membership Monitor (CMM) heartbeat
- Interface Definition Language (IDL) calls to remote CFS objects
- Data transfer between nodes
- Network traffic from nodes hosting global interfaces[8]

When all I/O is local to each of the nodes, very little traffic will be present on the private interconnect. However, when there are services requiring remote I/O, either because they are scalable services or due to failure of some kind, then network throughput and latency will impact performance.

7. Diskset is a Solaris Volume Manager term referring to a collection of disks. Disk group is the equivalent Veritas Volume Manager term. Diskset is used throughout this document to imply both terms.

8. A Global Interface (GIF) hosts a global IP address that is visible to all nodes hosting scalable services.

For performance reasons, it is recommended that Gigabit Ethernet be used as the private interconnect infrastructure. The underlying kernel drivers will multiplex the various messages across all available interfaces using a round-robin algorithm. With the current range of Sun Enterprise™ servers, it is unlikely that the throughput of a pair of Gigabit Ethernet interconnects will ever limit overall system performance, although large Sun Fire™ system configurations may benefit from multiple interconnects.

Placing Application Binary, Data, and Log Files On CFS

The global file service offered by Sun Cluster 3.0 software provides system administrators greater flexibility and simplified management of application binary and log file location when compared to Sun Cluster 2.2 software. These files can now be placed on either a file system local to each cluster node, or a cluster file system. However, there are few reasons to install anything on individual nodes, as this only complicates subsequent management of the application and its configuration files.

The pros and cons for each approach are:

TABLE 1 Benefits of using the global file service for application executables.

Installing on cluster file systems	Installing on local file systems
Patches and upgrades need to be done only once	Patches and upgrades need to be done multiple times
Changes to configuration files need to be done only once	Consistent changes need to be made to multiple copies of the configuration files
Care must be taken not to accidentally delete any program or configuration files, as this may stop the application from working	Accidental deletion of program or configuration files may prevent the application from working on a one node
Upgrades that overwrite the original executables may interfere with the correct operation of the running application	Upgrades can overwrite local copies of application executables when the application is not running on that node, without affecting the original version of the application running on an alternative node

Scalable services, or applications that run on several nodes concurrently, may present an additional problem when considering where to place the log files. Typically, these applications are not designed to have multiple instances sharing the same executables, so they offer no locking mechanism to handle concurrent access to log files. In these situations, it is still possible to locate the application executables on the global file service. Under these circumstances, log files should be created as symbolic links to a location on a local file system.

Although Oracle Parallel Server is a scalable application, because it runs on more than one node concurrently, it directs access to its control and log files, so no additional measures are necessary. When installing Oracle Parallel Server, it is safe to install the executables onto the global file service, because the data, log, and control files are managed on raw disk by the Oracle software itself.

Impact of Changing /Etc/System Parameters

As the Solaris Operating Environment has developed and matured, there are fewer reasons to tune system parameters in `/etc/system`, as most of the kernel structures referred to by the variables have become either obsolete or self-tuning. However, a few variables are still frequently modified, with good reason. Table 2 lists variables with restricted tuning when running Sun Cluster 3.0 software. Tuning other variables should create similar effects to those on “vanilla” Solaris software, with the understanding that tuning under Sun Cluster 3.0 has not been tested as thoroughly.

TABLE 2 The impact of tuning kernel (`/etc/system`) variables.

Kernel Variable Name	Description	Impact
ncsize	Size of the Directory Name Lookup Cache (DNLC)	An increase in ncsize will increase the numbers of CFS files in memory at any one time. Since CFS files consume far more memory than those from local file systems, this results in serious performance degradation of the system.
maxusers	Multiplier for several kernel structures	Maxusers has an affect on ncsize. Since increasing ncsize is not advisable, if maxusers is increased, ncsize should be explicitly set to its nontuned value.

Application-Specific Setup Tips

Use of CFS does not preclude access to any of the Solaris software performance-enhancing features such as kaio, aio, directio, large pages, and so on. In most cases, the application-specific sections (below) simply reiterate and summarize the suggestions that would apply on a standalone server. For a more detailed discussion of application tuning and capacity planning, see Resources.

The sections below cover HA Oracle and Oracle Parallel Server (OPS), Web services including iPlanet™ Web Server and Apache Web server, NFS, and other data services.

Oracle

Oracle supports two distinct modes of operation within Sun Cluster 3.0 software: HA Oracle and OPS. As with any database implementation, it is important to understand the I/O pattern that will be generated, as this guides the choice of logical-to-physical mappings made when laying out the database.

The patterns for online transaction processing (OLTP) and decision dupport dystems (DSS) applications are radically different. OLTP needs substantial in-memory cache (system global area or SGA) for frequently used data pages, and generates relatively small amounts of random I/O. DSS systems need large memory sort areas and generate substantial amounts of sequential I/O. In both cases, the ratio of disks to controllers is the most important factor, to ensure there are no physical bottlenecks.

Clustered file systems configured for HA Oracle data and log file use should always be local to the node running the service. So, while it is perfectly acceptable to configure a single cluster file system for Oracle executables, separate disksets should be configured for data and logs of every HA Oracle service. This provides the flexibility to ensure that data access will remain local after a failover, whether it occurs due to administrative intervention or configured resource group dependencies.

File System or Raw Device?

The next choice is whether to use raw disk or file systems for the Oracle data. Many database and system administrators shy away from using raw disk because of its perceived additional complexity. However, raw disk offers the highest performance when used with a tuned SGA, as raw disk has the shortest kernel code path.

If the use of file systems is preferred, then UFS mounted globally with the `forcedirectio` option offers the next-best performance. The `forcedirectio` option impacts any other application whose data is collocated on the same file system, as none of the data will be cached in the UNIX page cache.

Using multiple file systems not only allows flexibility in the use of `directio`, but may also speed up fail-over time, because the Sun Cluster 3.0 HA infrastructure can recover underlying kernel structures for file systems in parallel. The only drawback to this approach might be that of ongoing space management.

Oracle Tuning Parameters

The same tuning rules apply to Oracle parameters for UFS mounted globally or raw global devices, as to nonclustered devices. The basic rules of thumb are:

- For DSS workloads, `db_block_size` should be as large as possible (16 Kbytes for Oracle 8.1.x) and `db_multi_block_read_count` set to 64, to allow one-Mbyte reads. A large Shared Global Area (SGA) is not particularly important, as very little data is reused.

- For OLTP workloads, `db_block_size` should be set to two or four Kbytes, and `db_multi_block_read_count` set to 16, to avoid reading in potentially unnecessary data. The size of the SGA should be such that the cache hit ratio is greater than 90 percent.

Notes on Using Oracle Parallel Server

Although OPS uses raw disk[9] for storing data, it uses the file system to hold archive log files. The global file service of Sun Cluster 3.0 software helps considerably in simplifying the storage of these archive logs compared to Sun Cluster 2.2, but there is a corresponding overhead price to be paid.

Whenever a log file is archived, it must be written to a location on the global file service. For one node this will always result in the performance of remote I/O operations and, if the log files are large or frequently produced, may put an unacceptable demand on private interconnects.

An alternative approach is to create a separate archive location for each instance using two separate disksets mounted globally as two directories, for example `.../arch1/` and `.../arch2/`. Each directory has its primary path on the local node, and in the event of a failure that path is moved to the remaining node. This ensures that I/O is always kept local.

When choosing the private interconnect for an OPS system, it is advisable to pick the fastest network interface card (NIC) available.

NFS

Currently, NFS is a failover service, not a scalable one. Therefore, the configuration rules are identical to a Sun Cluster 2.2 system. This means that in order to fully utilize the outbound NIC bandwidth, one or more NFS resource groups should be configured per cluster node. As a two-node example, breaking up home directories into alphabetical batches (a-l*, m-z*) would be sufficient.

On large NFS configurations, it may be appropriate to further break the shares up into several smaller units, as these can be recovered in parallel should the resource group need to be failed over. Each of the shares has to be configured as a separate diskset with its own file system. The downside of this approach is space management.

9. Only the CVM functionality of VxVM can be used for storing these raw devices. The single exception is when the A3500 is used, because it does not require the additional volume management product.

iPlanet and Apache Web Servers

Both the iPlanet and Apache Web servers can be configured as failover or scalable services. The latter offers the benefits of faster response time to HTTP requests, and greater scalability where appropriate packet dispatch table settings are used.

Web server binaries can reside on the cluster file system, but their log files should be symbolically linked to a file in the local `/var` file systems. This is because these Web servers were not designed to manage concurrent updates to their log files.

When using a scalable Web service, between one and seven of the nodes will be remotely accessing the cluster file system, but not using its primary path. This rarely presents a problem, unless the cluster serves up very large, infrequently used (hence uncached) data files. An example of this might be a large multimedia library. Under very heavy loads, it may be necessary to configure an additional private interconnect.

CHAPTER 5

System Management Considerations

This section discusses typical system management operations such as patch management and software upgrades and how CFS can be used to best advantage when performing these tasks. However, best practices for backup and restore are outside the scope of this paper, because there are so many possible implementation options.

File System Management and Other Software Issues

Despite being an infrequent operation, increasing the space available in a file system is an important management task that should be completed as quickly as possible to ensure that the system does not grind to a halt when the last few disk blocks are allocated. Once again, the main considerations is to colocate the primary I/O path with the node on which the `growfs` command is executed.

Similarly, when applying patches to software located on the global file service, assuming the appropriate change control procedures have been followed, the fastest way to implement the changes is to apply the patch from the node that hosts the primary I/O path.

CHAPTER 6

Capacity Planning for Sun Cluster 3.0 Software

A key concern for system administrators ise whether the global file service or any of its associated features have an impact on capacity planning models. Such features are never free, inevitably consuming additional memory and CPU, as well as demanding appropriately configured private interconnects in order for the cluster to perform.

Calculating Memory Overheads

Sun Cluster 3.0 software comprises of a collection of kernel modules, user-level processes, and daemons. Additionally, every active file (files currently in use or active by virtue of having been cached by the Solaris Directory Name Lookup Cache) has extra kernel memory structures to manage coherent caching of their attributes and data across the cluster. These additional structures exist on the CFS client, CFS primary, and each CFS secondary.

To calculate memory overhead, a figure of approximately 22 Mbytes was generated by booting a cluster with the basic cluster infrastructure running and no services started, and comparing it with an identical server loaded with Solaris 8 software only.

The overhead associated with active files can be calculated by opening a number of files on one node, and reading the first byte of these files on a second node.

Other than these two quantities, there are no additional factors that must be considered when planning a Sun Cluster 3.0 system, beyond those that would apply to a standalone server.

Similarly, the heuristics used with monitoring tools that gather system statistics remain the same. CFS is integrated directly with the Solaris software page cache, so `vmstat(1M)` is still the most appropriate Solaris tool for tracking down memory shortages.

Sizing the Private Interconnect Network

It is unlikely that the cluster private interconnect will be a bottleneck to system performance on Sun Enterprise servers if the guidelines for configuring a cluster given in this document are followed. Only Sun Fire servers with more than eight 750-MHz UltraSPARC™ III CPUs would have the capacity to drive six Gigabit Ethernet private interconnects, let alone do any application processing. However, if there are good reasons why substantial amounts of remote I/O are needed, it is useful to know what the practical limits are in terms of achievable network and CPU cost in packet transfer.

Assuming that an application is tuned and well behaved, the I/O throughput will be initially governed by the underlying disk subsystem. If it has been reduced or removed, the next limitation to remote I/O will be the size of the network pipe. The access mechanism for Ethernet limits the achievable throughput to about 60 percent of the peak bandwidth. Thus, a 100BaseT link carries a maximum of around seven Mbytes per second of traffic, and a Gigabit Ethernet network can carry approximately 70 Mbytes per second.

It is worth stressing that adding more private interconnects will not improve the performance of a single I/O stream. So, if an application is writing to remote storage via CFS at one Mbyte per second, an extra 100baseT link will make no difference in performance.

The Impact of Increased Node Separation on CFS Performance

The data given in above are for cluster nodes connected back-to-back with short (less than ten meters) cables. In many cases, this is how systems are deployed in data centers. However, they are not always deployed this way, and the ramifications of separating them by large distances is described below.

As interconnect length increases so does communication latency. Although the increase is small, on the order of five nanoseconds per meter[11], large-node separation will cause this to become a significant number. A node separation of five kilometers, for example, will addition 25 microseconds to each TCP-IP message sent over the private interconnects, and may cause a drop in I/O performance.

11. This is due to the finite speed of light in glass of the fiber.

Optimal Disk-to-Node Allocation in Clusters with More Than Two Nodes

Sun Cluster 3.0 software supports three topologies for disk connection: clustered pairs, pair + M, and N+1. Coupled with CFS primary/secondary I/O channel architecture, this brings up other issues that should be considered when building clusters with more than two nodes.

If the application sends a large number of file creation, deletion, and write requests, it will perform best when the primary I/O path is local. For architectures such as pair + M, this restricts the locations where the application could sensibly reside to just the 'pair' nodes.

If the application is read-mostly, as in Web or application servers, then any of the above architectures are suitable, as only small amounts of disk I/O and all the global networking traffic are passed across the private interconnects. It should be noted that if the public network is a Gigabit Ethernet, then private interconnects should also be Gigabit Ethernet, because otherwise packet distribution between nodes may bottleneck.

CHAPTER 7

Disaster Recovery

Customers are often tempted to combine — or even confuse — disaster recovery with high availability, when in reality these address different objectives. The aim of a disaster recovery solution is extensively covered in *"Blueprints for high availability"* by Marcus and Stern (see the Resources section), and will not be repeated here.

Disaster recovery solutions are usually constructed in two ways. The first, which is not really a disaster recovery solution but rather protection against problems associated with a single location, uses extended connectivity. This requires the use of long-wave Gigabit Interface Converters (GBIC) and nine-micron fibre infrastructure. Although it protects against catastrophes at a single location, such as fire or flood, it does not protect against data corruption or deletion, accidentally or otherwise.

If this is sufficient for a customer's disaster recovery policy, then only the performance ramifications highlighted in the previous section need be considered before implementing such a solution.

In the long term, a more appropriate solution is to use a combination of Sun's Network Data Replicator (SNDR) and Instant Image products. If an RDBMS is involved, then the database replication technology available within the specific database product should be used. Other third-party database replication products are available. All these have better performance characteristics and recovery facilities than the use of CFS over extended distances.

CHAPTER 8

An Application Developer's Guide to Writing HA Applications That Use CFS

This section outlines the best practices for using CFS as a data repository for HA applications.

CFS Usage Basics

Note that the points below are clarifications of files ystem semantics in UNIX, and are not specific to CFS.

CFS is consistent with respect to processes on multiple nodes simultaneously accessing files in it:

1. Process P1 calls write(2)
2. Process P1 gets back a successful return from write(2)
3. Process P2 calls read(2)

If access to a file is completed in the order specified above, then process P2 will get back data that process P1 just wrote to the file. This remains true, even if the two processes are on different nodes.

Note that if the call order is:

1. Process P1 calls write(2)
2. Process P2 calls read(2)
3. Process P1 gets back a successful return from write(2)

In this case, process P2 could have retrieved data before or after the write call was made.

Multiple-reader, single-writer semantics are maintained per file.

CFS does not serialize access to a file by multiple readers. It does, however, serialize access to a file by multiple writers, even if the writes are being done to different parts of the file. In other words, any time a write is being done to a file, all other calls to `read(2)` and `write(2)` are blocked until that write is completed.

Normally, writes to the file system using the `write(2)` system call do not make it to disk synchronously. Applications may choose among the following semantics:

- Send every write to disk. Applications can do this by using the `O_SYNC` flag to the 'open' system call, or by using the file in `directio` mode — see `directio(3C)`. This ensures that future calls to `write(2)` make it to disk before returning to user space. Clearly, this has a serious performance impact.
- Make consistent snapshots of the file at sync points using `fsync(3C)`. In this case, the pattern of file system usage is a series of writes followed by a call to 'fsync'. When the `fsync(3C)` call returns, all previous writes are guaranteed to be written to disk. Depending on the number of `write(2)` system calls between `fsync(3C)` system calls, this could lead to much better performance compared to the use of `directio(3C)`.
- Never sync out the data. This is a sufficient approach if the application is only writing out journalling or logging information, which is used for gathering noncritical statistics. In this approach, if the node on which the application is running crashes, some of the data will not be written out, so the data cannot be critical.

Note that in NFS, a `close(2)` system call flushes data to disk — this is not true for CFS.

Another way to access CFS is to use `aiowrite(3AIO)`, and then use other `aio` routines to provide notification of the write completion. The benefit is that the write is asynchronous. In a carefully written application, the response to the client may not need to wait for the write operation to complete. Note that in Solaris 8 software, `aio` to files are done via user-land libraries that run as different threads in the context of the process.

Directory operations on CFS are typically much slower than the same operations on the underlying file system, because CFS needs to sync out directory updates to maintain HA semantics. This is expected to improve in future versions of CFS, but for now, there is little that can be done about it except to avoid directory operations in time-critical paths.

Usage Best Practices by Application Profile

A generic guiding principle is to write the application in such a way that file access times are minimal at the point where a client request is waiting for a response. Typically, this means that the `read(2)` and `write(2)` system calls finish quickly. To this end, applications should be written keeping these guidelines in mind.

Single Reader

- The process doing the reads should be colocated with the CFS primary. This will speed up noncached reads, and reduce traffic on the cluster interconnect. See the Sun Cluster man pages and documents at `SUNW.HAStorage(5)` for details on how to achieve this.
- Sequential reads are better for performance than random reads, because in the sequential-read case, CFS does read-ahead caching.
- Where possible, the application should precache the data it requires to respond to client requests.
- On the node where the application is running, memory size should be large enough to hold a working set of file data.

Multiple Readers

- See guidelines for single-reader applications.
- If one of the readers is running on a node that is colocated with the CFS primary, then noncached reads from that node are faster than from other nodes. Cached reads are the same speed.

Single Writer

If possible, applications should be written in such a way that spaceis preallocated for data files in the file system. Strategies include:

- Do not create sparse files by using `lseek(2)` to seek past the end of the file, and then writing to the holes. Instead, preallocate space in the data file by writing zeroes to it or using `mkfile(1M)` before writing to it. This will provide a performance boost at the time of the write.
- Applications using the `append` mode of files to write data should instead preallocate the file and use `lseek(2)`. Note that this requires keeping track of the end-of-data pointer, since it is no longer the same as end of file.
- If possible, applications that need to maintain log files should use a circular log file, keeping only recent data in the log.

Nonallocating Writers

- Similar to the single-reader case, keep the writer collocated with the CFS primary.

Allocating Writes (Appends, Sparse Files)

- It is recommended that the CFS file system used by the application should be mounted without the `syncdir` option if it is a UFS file system. Write the application to handle `ENOSPC` errors from `close(2)`.

Single Writer and Single / Multiple Readers

- See the guidelines for single writer.
- Try breaking up the file into smaller files, with the aim of reducing contention for the file to be written to — in other words, structure this so that most of the time, some files fall in the multiple-reader category and other files fall into the single-writer category. The higher the contention for files that need to be written to, the worse the performance.
- If the application causes a writer to do a series of writes before it is done, use file locking (`see fcntl(2)`) to synchronize access at the application level between readers and writers. This will ensure better performance for the series of writes, by preventing contention for data pages between the writes.

Multiple Writers / Readers

- See the guidelines for single writer and single / multiple readers. Breaking up the file into multiple, independent files is even more important here.

CHAPTER 9

Appendix A: NFS versus CFS

CFS	NFS
Has a global namespace. Only one mount command is needed, with all nodes having equal access.	Multiple mount commands are needed, one per node. Access characteristics can differ from node to node, and depend on the `share(1M)` options.
Data caching is fully consistent.	Applications could read stale data for a short period of time.
Attribute caching is fully consistent.	Applications could read stale attributes for a short period of time.
File locking semantics are maintained across failover boundaries.	Programs could get SIGLOST under certain circumstances.
Does not support automounter mounts.	Supports automounter mounts.
Usage is possible only within the cluster. NFS is necessary to use this file system outside the cluster.	Usage possible by any machine on a network connected to this one.
Writes are cached until the regular Solaris mechanisms flush out dirty pages or the application syncs the data.	Writes are written through more aggressively to the NFS server.
Data pages are cached only by the CFS client, and not by the underlying file system on the CFS server mode.	Data pages are cached by both the NFS client and the underlying file system on the NFS server node.
Supports seamless failover. Failover is faster.	Seamless failover in the nonclustered case available only for read-only services.
Designed to exploit future fast cluster interconnects using remote DMA and zero-copy functions.	Not designed for such usage.
Provides a mechanism to support global devices.	Does not support remote devices.

CHAPTER 10

Resources

Additional information can be found at:

- *Sun Cluster 3.0 Architecture* white paper: www.sun.com/software/whitepapers.html#cluster.
- *Sun Cluster 3.0 U1 Concepts Guide (Part #806-7074)*: docs.sun.com

References

- Wong, B.L., *"Configuration and capacity planning for Solaris servers,"* 1997, Prentice Hall Publishing, Upper Saddle River, New Jersey.
- Cockcroft, A. and R. Pettit, *"Sun performance tuning: Java and the Internet,"* 1998, Prentice Hall Publishing, Upper Saddle River, New Jersey.
- Marcus, E. and H. Stern, *"Blueprints for high availability: Designing resilient distributed systems,"* 2000, Wiley, New York, New York. www.wiley.com.

About the Author

Tim Read is a lead consultant for the High End Systems Group in Sun's Joint Technology Organization in the U.K. Since 1985, he has worked in the U.K. computer industry, joining Sun in 1990. He holds a BSc in Physics with Astrophysics from Birmingham University. As part of his undergraduate studies, Read studied clusters of Sun systems; now he teaches and writes about Sun Cluster systems and software.

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303

1 (800) 786.7638
1.512.434.1511

www.sun.com