The architecture of the HP StorageWorks Enterprise File Services Clustered Gateway white paper

Implementing the HP Clustered File System

Introduction 3

Issues in cluster file system design 4
  Master server–based architectures 4
  Performance and availability limitations 5
  Manageability limitations 6
  Cache management limitations 6
  Summary 7

HP breakthrough: A fully symmetric cluster 7
  Distributed Lock Manager (DLM) 8
  Symmetric metadata access 8
  Cache coherency across nodes 8

HP Clustered File System overview 9
  Clustered File System locking infrastructure 9
    Distributed Lock Manager 9
    Lock Caching Layer 10
  HP Clustered File System and Journal 10
  Virtual Device Layer 11

The HP Clustered File System availability and recovery infrastructure 11
  Cluster node health 12
  Network connectivity monitoring 12
  Application/middleware monitoring 12
  Device monitoring 12
  Application restart and failover 12
  Storage monitoring and access control 13
  File system recovery 14
  Administrative logs and diagnosis tools 14
  Multi-path I/O 15

Clustered File System summary 15

Other cluster file systems 15
Red Hat GFS 15
VERITAS SANpoint Foundation Suite HA 15
IBM GPFS 16

Conclusion 16

For more information 17

Introduction

A clustered file system allows multiple servers to share files and data while maintaining read/write consistency and thus data integrity. Cluster file systems are at the heart of emerging data center architectures in which multiple, highly cost-effective servers handle mission-critical file systems that previously required expensive and/or proprietary servers. In this way, cluster file systems play a much more central role in the emerging data center than single-node file systems have played in the past.

Cluster file systems provide a necessary foundation for fulfilling the central promise of storage area networking—the ability to consolidate enterprise data into unified pools, simplifying management and improving flexibility and availability. To be useful for the enterprise data center, however, such a file system must meet a set of stringent technical requirements, including:

- Compatible with existing network architecture providing single-node semantics across multiple servers.
- Supportive of commercial or enterprise workloads, including intense read/write activity and heavy volumes of file creation and deletion.
- Scalable, to permit multiple servers to work effectively together.
- Completely recoverable, able to survive hardware or network failures while preserving data integrity.
- Easy to install, configure, monitor, and maintain.

While the need for such capabilities is obvious, developing the solution is extremely complex and requires overcoming difficult problems in basic computer science. A fundamental problem at the heart of the challenge is how to maintain state consistently across many servers, while allowing this state to be updated from any of those servers and with complete recoverability. Achieving this goal is far from easy.

As a result, the promise has gone unfulfilled. Most efforts have failed entirely or fallen so significantly short that they are useful only in niche roles (for example, pre-production print image sharing).

HP has introduced the HP StorageWorks Clustered Gateway with HP StorageWorks Clustered File System (CFS) Software that finally delivers on the promise of true cluster file systems. With its symmetric, fully recoverable design, CFS has been designed from the start to meet enterprise requirements for scalability, availability, and reliability. It is easy to install and administer, scalable yet fully recoverable, and supports single system semantics for both performance and application independence.

This paper reviews the shortcomings of previous clustered file system approaches and shows how the CFS architecture overcomes the fundamental design limitations that hobbled earlier attempts.

Issues in cluster file system design

Previous efforts have not produced cluster file systems that can meet the needs of emerging architecture of the enterprise data center. Their failings spring from the architectural approaches they have adopted. These can be broadly divided into two categories:

- Master server–based designs (metadata servers)
- Scientific computing file systems

Both of these approaches have major architectural limitations from the perspective of the data center requirements previously described. Master server–based designs suffer from inherent scalability and performance limitations, in some cases lack full recoverability, and can be complex to implement.

Efforts in the second category may provide a high degree of scalability and performance, but only for scientific computing workloads. That means large, sequential reads and writes, few random reads and writes and very few updates, little read/write sharing, no transactional requirements, relatively infrequent metadata operations (file creates, lookups, and so on), and little emphasis on recoverability and application availability. These design goals differ dramatically from the data center requirements previously described. Thus even scientific computing file systems that are quite successful in their own domain are unsuitable for data center use.

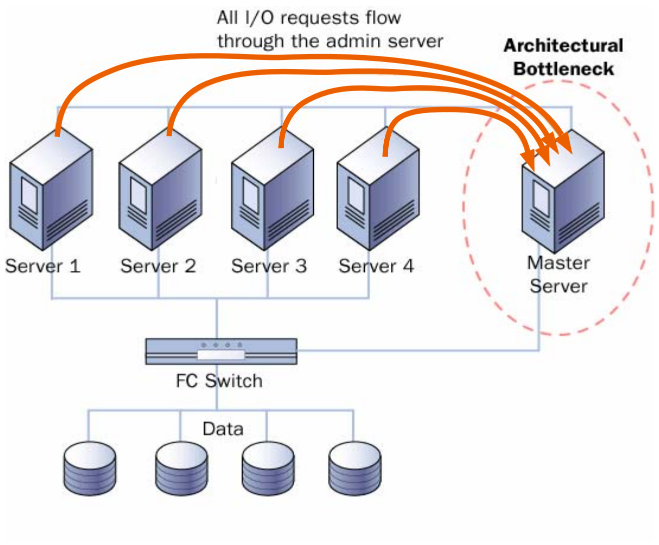
Since the master server–based designs are at least intended for use in the data center, this paper focuses on those. The following sections describe this approach. A later section discusses specific competitive products in light of these architectural schemes.

Master server–based architectures

Many existing clustered file systems have emerged from traditional high-availability solutions. Typically, these solutions were developed originally for database applications, where high availability could be achieved by deploying a pair of servers in a primary/backup, active/passive configuration, with dual-headed SCSI storage connected to both. At any instant, only one server would access the storage, but in the event that the primary server failed, the backup would take over the storage and commence serving application requests. Over time, this architecture was stretched to allow more than two servers to be clustered together—maintaining the assumption, however, that each storage resource was used by only one server at any instant.

An obvious step for such a system was to provide simultaneous access to storage through a cluster file system. The existing file systems used in these environments, however, assumed that no concurrent access could occur: all of the file system data structures (metadata), and the code that uses and updates those data structures, relied on the fact that only one server was involved. Rather than start from scratch, the solution adopted was to specify that one node, the master server, should perform all file system metadata manipulation: all other nodes would forward these tasks to the master for processing, while data blocks pass over the storage area network (SAN) (see Figure 1).

Figure 1. Master server–based architecture

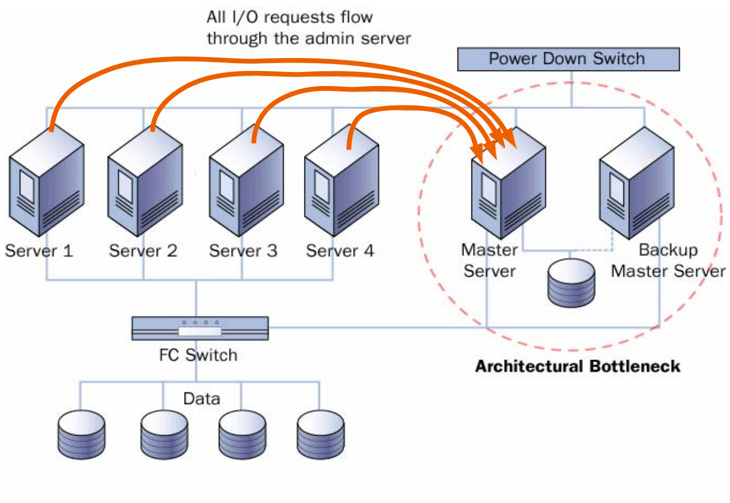


Performance and availability limitations

However, this architecture suffers a fundamental performance limitation. The master was required to manage all I/O traffic (consulting file system metadata to locate and create files, traverse directories, and so on), and thus quickly became a bottleneck, especially for write-intensive applications. The more servers contending for the master server's attention, the slower the system would operate.

Furthermore, because access to the file system was completely dependent on the master server, availability and recoverability were compromised. To address this problem, a backup master server was introduced, which could take over for the master server in the event it failed (see Figure 2).

Figure 2. Master server–based architecture with backup



Introducing a backup server addressed the vulnerability of the architecture to the failure of the master, but availability could still suffer. In the event of a failure, the failover process took a significant amount of time. First, the backup server would need to ensure that the primary was not still using the file system. Then it would mount the file system and perform appropriate recovery operations to bring the file system data structures into a consistent state. Finally, the client servers would rebind themselves to the new master server.

Manageability limitations

At the same time, the backup server introduced increased complexity. The administrator now confronted three server configurations: the master, the backup for the master, and the client servers. The high-availability failover mechanism used between the master and its backup itself required further configuration and in some cases even extra hardware, such as a network-controlled power switch used to ensure that both do not attempt to access the file system data structures simultaneously.

All of this did nothing to address the performance bottleneck inherent in the master server design, or to address a subtler problem that merits a discussion of its own.

Cache management limitations

In any file system—single-node or cluster—a key to performance is the use of caching to speed data access. Most requests to read data can be fulfilled using previously read data in cache, and write performance can be improved by deferring sending updates to disk, storing them in cache in the meantime instead, to coalesce multiple requests (unless an application specifically requests otherwise). All of this is standard practice for single-node file systems, where a portion of server memory is used to cache data read and written by applications on that node.

In the clustered environment, however, each server has its own cache. A question arises: how are these to be kept consistent? If they are not, applications may be given incorrect, out-of-date data: an update could be performed by one server, while other servers, having already cached the data in question, continued to use that outdated information.

One approach, which some cluster file systems have adopted, is to avoid the question by turning off caching entirely. This exacts a steep performance penalty for almost all workloads, however, since it forces all requests to go directly to disk and forfeits the performance advantages that led to the universal adoption of caching in the first place.

The alternative is to introduce a locking mechanism. A lock is a data structure that controls what a server is permitted to do with a particular piece of data. For example, it might either permit multiple servers to hold a copy of a piece of data for reading only, or might permit a single server to perform an update to that data. By using locks to coordinate all cached data and all updates to that data, the cluster can ensure that before any update is performed to a particular piece of data, it is removed from any other node's cache. In this way, consistency is guaranteed.

A major design decision, then, becomes how to manage locking across multiple nodes. Different cluster file systems implementations have adopted different strategies. A simple-minded approach is to require administrators to manage locks themselves, manually assigning the ability to update particular files to the appropriate node every time an update is required. This obviously imposes severe limitations on the applicability of the file system.

The most common design adds lock manager functionality to the master server: in this case, every other node sends requests to the master server to obtain appropriate locks for the data in its cache, just as all it does to request that file system data structure operations be done on its behalf.

Unfortunately, of course, this approach increases the load on that master server, and it adds another piece of functionality that must be failed over to assure high availability. Limitations on cache performance remain a significant challenge for master server–based cluster file systems.

Summary

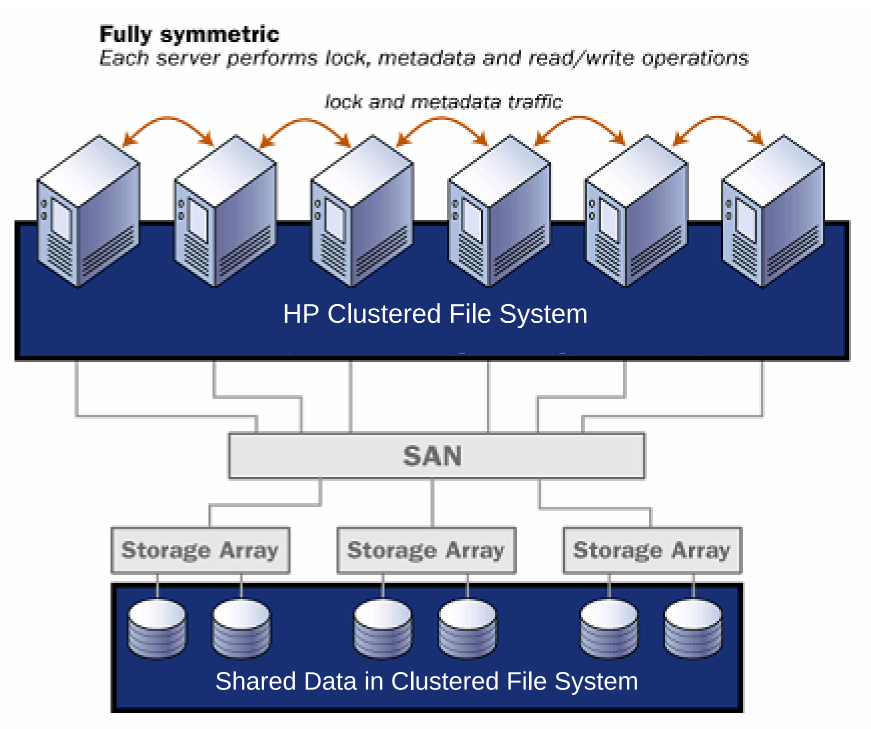
Cluster file systems based on master servers—whether for metadata operations, locking operations, or both—are relatively easy to implement as an extension of traditional, single-node systems. However, they suffer from performance and availability limitations inherent in the master server's singular role. As servers are added, the load on the master server increases undercutting performance and subjecting more nodes to loss of functionality in the event of a master server's failure. By contrast, an appropriately designed cluster file system should be able to exploit multiple, independent servers to provide better scalability and availability, insulating the cluster from any individual node's failure or performance limitation. The next section will describe just such an approach.

HP breakthrough: A fully symmetric cluster

The HP StorageWorks Clustered File System overcomes the problems highlighted by the previous discussion and provides the enterprise data center with a scalable, fully recoverable, high-performance, clustered file system. With the CFS software, the HP StorageWorks Clustered Gateway is designed to be easy to install, configure, and manage.

The key design principle behind CFS that makes this possible is symmetry among the nodes: the tasks that the cluster must perform are distributed uniformly across its members, enhancing scalability and availability. In the example shown in Figure 3, both metadata and lock management tasks are spread across all six nodes in the cluster. The following architectural overview explains how a symmetric cluster file system approach eliminates the administrative complexities and the performance and availability issues with previous clustered file system approaches.

Figure 3. HP symmetrical architecture


Fully symmetric
Each server performs lock, metadata and read/write operations
lock and metadata traffic
HP Clustered File System
SAN
Storage Array
Storage Array
Storage Array
Shared Data in Clustered File System

Distributed Lock Manager (DLM)

The fully symmetric DLM is the subject of multiple patent applications and is a crucial element in the scalability that CFS provides. The DLM guarantees proper locking behavior without relying on a master lock server. Each server participates in coordinating locks across the cluster, and the load is spread evenly in such a way that no node becomes a bottleneck. In the event that a node fails, responsibility for the locks it had been coordinating is redistributed to other, surviving nodes. Even cascading failures, where multiple servers leave the cluster unexpectedly, can be handled.

Symmetric metadata access

Similarly, all file system metadata operations can be performed symmetrically by every server in the cluster. This eliminates the metadata master server performance bottleneck and also provides improved availability because the cluster need not wait for a backup master server to come online in the event of failure. The CFS implementation has sophisticated mechanisms to allow surviving nodes to recover from other nodes' failures—even multiple, cascading failures—and to continue to work.

Cache coherency across nodes

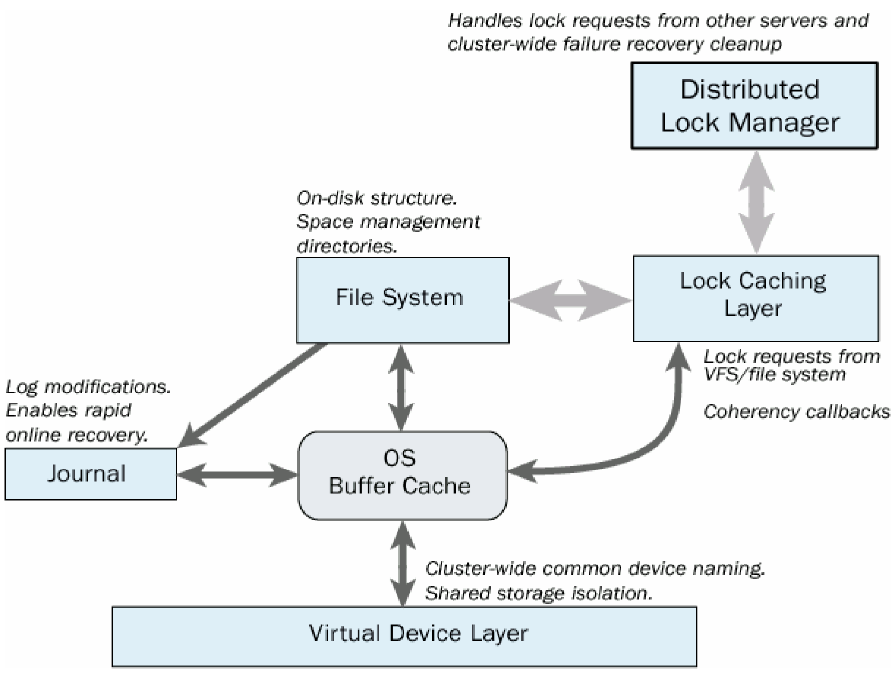
The Clustered Gateway supports full cache coherency among all nodes in the cluster. Both reads and writes can be cached, as on a standard single-node file system. Every read is guaranteed to retrieve the most current data, even if that data has been written moments before on another node.

Updates can be maintained in a server's cache, with multiple or overlapping writes being coalesced for better performance. The CFS uses a "copy back" strategy for data written to cache. That is, the data need not be written to disk immediately (unless, of course, the application requests that it be). In the event that another server requires the data to satisfy a read request, then the updates will be written out to disk; otherwise, the operating system, as it does for single-node file systems, will schedule the writes after an interval of time. The trigger for copying updated data back to disk is an incoming lock request covering that data arriving from another node. The tight integration between the DLM and the caching mechanism is the subject of the next section.

HP Clustered File System overview

Figure 4 provides an overview of the CFS components. The next sections discuss various aspects of these components in detail.

Figure 4. HP Clustered File System overview


Handles lock requests from other servers and cluster-wide failure recovery cleanup
Distributed Lock Manager
On-disk structure. Space management directories.
File System
Lock Caching Layer
Lock requests from VFS/file system
Coherency callbacks
Log modifications. Enables rapid online recovery.
Journal
OS Buffer Cache
Cluster-wide common device naming. Shared storage isolation.
Virtual Device Layer

Clustered File System locking infrastructure

CFS locking infrastructure consists of two principal components: the DLM and the Lock Caching Layer.

Distributed Lock Manager

The DLM runs on each node in the cluster. It provides locking primitives that allow data to be shared in a read-only fashion across arbitrarily many nodes within the cluster or updated by a single node at a time. Each file system within the cluster has its own independent domain of locks, and there are separate domains used to coordinate cluster-wide activities that span multiple file systems, such as adding a node or reconfiguring the cluster configuration.

As previously mentioned, responsibility for coordinating proper lock behavior is spread across all the nodes of the cluster. Also important is the fact that while the DLM ensures that all nodes have consistent information about the state of locks—for example, it prevents one node thinking it has an exclusive lock to update a piece of data, while other nodes still retain a lock covering read access to an earlier version of that data—it does this without employing broadcast messages sent to all members of the cluster. Using broadcasts would simplify the task of maintaining consistency, but at the cost of undermining performance and scalability: as cluster size grows, network traffic would explode.

Instead, using carefully tuned and validated locking protocols, a lock can be acquired and then passed from server to server while involving only those nodes that have an interest in that particular lock, plus perhaps one other node. This other node, the lock's "home node," is responsible for coordinating activity involving that particular lock. Because the full set of locks has their home nodes spread evenly across the cluster, and because the set of nodes interested in a particular item of data typically varies from one item to another, no single node becomes a bottleneck.

The DLM has a key role to play in recovery from server or networking errors. Often, the failure of a node to reply to lock manager messages is the first indication of its failure. Regardless of how a failure is detected, the DLM is deeply involved in orchestrating the actions of surviving nodes as they recover from another node's failure.

Lock Caching Layer

Just as data benefits from caching, because often an item of data that is read or updated now will be read or updated again in the near future, so too do locks. Frequently, when a server requests a lock, it may have held that very lock in the recent past. In fact, often no other node will have used the lock in the intervening period. Thus performance can be enhanced by having servers retain locks opportunistically—only releasing them when some other node makes a specific request, or, in some cases, after some period of time has elapsed.

The Lock Caching Layer performs this function in CFS. Reacquiring a lock that has been cached requires little additional overhead beyond a standard SMP operating system lock operation. Sizing of the caches are made based on physical memory in the server, but by carefully structuring lock data structures to be highly space-efficient, even small servers can maintain many thousands of outstanding locks in cache simultaneously. This maximizes the frequency with which lock requests can be satisfied internally.

HP Clustered File System and Journal

CFS's sophisticated locking infrastructure exists to serve the needs of the Clustered Gateway. CFS is a symmetric design, with on-disk data structures that have been optimized to allow multiple nodes within a cluster to perform metadata operations simultaneously. It is designed to handle commercial or enterprise workloads, including write- and metadata-intensive ones. Without HP Cluster Volume Manager, file systems can be as large as the native OS are supported, with no pre-allocated limit to the number of files, and with good performance even for directories with large numbers of files. With HP Cluster Volume Manager Software, large file systems of up to 16 TB can be supported.

To provide recoverability in the event of a server failure, all file system metadata operations are recorded in an on-disk journal. Journal entries are "snooped" across the cluster: if one node has performed an operation and committed it to the journal, then results of that operation will be visible to all other nodes in the cluster. However, for performance, servers may choose to delay committing operations, and writing the associated entries into the journal, until requested to do so by an application, or until another server indicates its need to use the affected data through the locking infrastructure.

Each file system contains a single journal shared by all nodes accessing it, providing simplicity and flexibility, since there is no need to pre-allocate journal space on a per node basis. Nodes are able to commit operations to the journal independently in any order, which prevents the single physical journal from becoming a performance bottleneck.

Virtual Device Layer

All accesses to storage from the CFS are handled by the Virtual Device Layer, which has two functions: it provides persistent, cluster-wide device names, and it allows access to shared devices to be controlled by the Clustered Gateway clustering infrastructure.

Cluster-wide device names are required if clusters consisting of many servers and many storage resources are to be manageable. In their absence, each server at boot-up time will enumerate the storage device Logical Unit Numbers (LUNs—a disk or a volume in a storage array) it can see over the SAN and assign disk device names sequentially, based on the order in which they are discovered. This means that the disk or volume known as /dev/sda on one server might be known as /dev/sdc on another, and /dev/sdh on yet a third. In principle, a given storage LUN could be known by different names on each node of the cluster. Even worse, these device names can change unpredictably over time, producing so-called "device slippage." Consider the device known as /dev/sda on the first server. If the SAN is reconfigured to make an additional LUN available, after a reboot what was formerly /dev/sda may now be known as /dev/sdb, the new LUN having taken over the name /dev/sda.

To avoid this chaos, the Virtual Device Layer provides persistent, administrator assignable, cluster-wide names. Instead of worrying whether /dev/sda refers to the intended resource, the administrator can choose to use the name /dev/psd/custdb1 across all servers.

The second function of the Virtual Device Layer is to control access to storage resources. The Virtual Device Layer will prevent any access conflicting with other concurrent uses: for example, it guarantees that no one can re-format a file system that is mounted on other nodes. In a single-node environment, this sort of safety is standard, and easy to implement, but in a cluster, it requires cluster-wide coordination. Similarly, the Virtual Device Layer will prevent the server from attempting to access a shared file system unless it is in communication with any other servers using that file system. It also integrates the Clustered Gateway's mechanisms to manipulate SAN access controls to arbitrate which nodes should be able to use a shared storage resource.

The HP Clustered File System availability and recovery infrastructure

In normal operation, the file system and supporting elements previously described serve to provide highly scalable and fully cache-coherent access to shared data. A key design principle of CFS, however, is that the system should behave correctly when failures occur, as well as in normal operation. Thus CFS architecture includes components that exist to detect various kinds of problems that may occur and to coordinate the cluster's response, ensuring that applications remain available and data integrity is preserved. These include:

- Node and network monitors
- Application/middleware, and device monitors
- An application/middleware restart and failover engine
- Fabric/storage monitoring and access control
- File system recovery sequencer
- Cluster-wide administrative logs and diagnosis tools
- Integrated support for multi-path I/O for Fibre Channel SAN redundancy

All of these components work together to provide full recoverability from a wide variety of potential failures, including simultaneous failures and cascading failures in which multiple cluster elements fail in quick succession.

Cluster node health

CFS provides extensive monitoring of node health and network connectivity. At any instant in time, the cluster maintains a list, agreed consistently cluster-wide, of which nodes are functioning properly. The cluster members exchange heartbeats to monitor each other's health.

In fact, internal protocols used go significantly beyond simple heartbeats. A group communications infrastructure underpins the CFS distributed algorithms. Group communications mechanisms provide the ability to maintain coherent state across multiple nodes, with systematic mechanisms for recovering and re-synchronizing state in the event of node or communications failure. In the CFS, this layer coordinates the cluster-wide agreed list of healthy nodes and also ensures that all cluster nodes agree on all aspects of cluster configuration. This makes it possible to administer the cluster as a single logical unit—an administrator can log in to any node and manage all of them—and it prevents many subtle problems that can occur in other environments where cluster members are configured on a node-by-node basis.

Network connectivity monitoring

The group communications layer also monitors network connectivity, perhaps over multiple independent network paths between servers. In the event that one network fails, the cluster can transparently fail over to using another network for cluster communications. The administrator can specify which networks should normally host intra-cluster traffic and which should only be used as a last resort in the event of failure.

Application/middleware monitoring

Beyond basic server health and network connectivity, CFS includes application monitors that can periodically probe application health. CFS includes a pre-built set for common applications, and a simple interface is provided that allows new monitors to be implemented for custom applications. An administrator can easily specify how often the probe should be performed, and how many negative results are required to declare the application failed.

Device monitoring

Similarly, CFS provides for "device" monitors that probe aspects of node health that might affect multiple applications on that machine. Device monitors are used to track accessibility of shared storage (as discussed in more detail in Storage monitoring and access control), and can be used to monitor nodes' memory or CPU usage. An interesting use of the device monitor feature is to probe a network gateway, ensuring that if a server loses connectivity to the outside world—and therefore, presumably, to the clients that use its services—its applications can be moved to a node that clients can reach.

Application restart and failover

CFS incorporates application restart and failover features. Regardless of the source of the problem, if a failure is detected, CFS can fail over affected file serving applications to another node, or, if the application seems to have hung but its server is functioning properly, the application can be restarted. The administrative interface allows failover policies to be easily configured: it is possible to have multiple servers configured to fail over to a common backup server, to cycle backup responsibilities (A fails over to B, B fails over to C, C fails over to A), or to implement more elaborate schemes. In addition, failover policies can be configured on a per-application basis, so if server A hosts two applications, one can be failed over to B and one to C if server A is lost.

CFS significantly improves application availability over many other clustering solutions. First, it makes it easy to implement application failover policies because all nodes can see all file systems simultaneously—so an application can be readily tested on a backup node, and very little reconfiguration is required to change the node on which an application should run. Second, CFS allows for fast failover times. Without a cluster file system, the first step when a server fails is to move its storage resources to a backup server, and start the process of recovering and then mounting the file systems on the new server, taking significant amount of time—minutes, or in some cases much longer (in some architectures, in fact, this may require rebooting the backup machine.) With CFS, the backup machine already has access to the necessary file systems and so the application can start rapidly once the cluster performs a brief recovery process. In some cases, the application can in fact be running on the backup node before the failure occurs, simply waiting for requests.

Storage monitoring and access control

As previously mentioned, CFS uses device monitors to track each server's ability to access required storage resources. Beyond that, however, CFS includes a Storage Control Layer that allows the cluster to track and control which servers have access to which resources. This is extremely important for guaranteeing data integrity and application availability in a cluster environment, in two respects: first, it provides the basis for presenting a single-system image of storage resources to administrators, which prevents "pilot error" that might cause data loss or application downtime, and, second, it protects the cluster in situations where cluster members are unreliable or lose network connectivity to one another.

First, the Storage Control Layer provides a single cluster-wide view of storage resources that protects the system from inappropriate and potentially damaging actions. Administrative mistakes have been found, in study after study, to be a prime cause of downtime, and a cluster environment's possible complexity raises the danger. A single cluster-wide view accomplishes a great deal in reducing the potential for error. The Storage Control Layer, working with the Virtual Device Layer on each cluster member, provides this. For example, as previously described, CFS implements cluster-wide device naming—allowing the administrator to rely on the fact that /dev/psd/custdb1 always refers to the same storage resource, and avoiding the likelihood of error when /dev/sdc means something different on each server. It also prevents all nodes from reformatting a file system that is in use by any node in the cluster; this reproduces in a cluster context a safety mechanism that is standard on individual servers.

However, the Storage Control Layer goes beyond what the Virtual Device Layer implements on each node, to ensure that storage resources are protected from inappropriate access from outside the cluster. Using SAN fabric access control mechanisms, CFS can either prohibit non-member machines from accessing cluster storage resources, or can raise an administrative alarm if such access has inadvertently been permitted. By doing this, CFS prevents a whole category of difficult-to-diagnose errors.

There is a second area where the Storage Control Layer contributes to system availability and data integrity. This concerns what happens when hardware unreliability—whether of servers or of networking—prevents cluster members from coordinating their access to shared data resources. As previously described in the discussion of cache coherence and locking, it is vital that cluster members be able to communicate lock state to coordinate their access to shared data structures within a file system. Consider a so-called "split-brain" situation, where a networking failure causes one group of servers to be isolated from the rest of the cluster, but where all the servers can still access the storage resources over the SAN. In this case, the cluster architecture must ensure that only one group will continue to attempt to use the shared resources.

HP CFS includes an innovative solution to this problem. The Storage Control Layer can use access control mechanisms in the SAN to control which servers are able to use which resources. In addition, a small, replicated, on-disk membership partition is used to allow groups of servers that are otherwise isolated from one another to communicate exclusively through storage. Thus, in a split-brain scenario, the two (or more) fragments of the cluster can discover one another through the membership partition, and in fact can arbitrate through this partition which fragment should continue to have access to the shared storage.

Storage control is an important and subtle area of cluster file system design. The HP approach, and how it contrasts with others, is discussed in greater detail in the HP white paper *Data Integrity with HP's StorageWorks Clustered File System*.

File system recovery

When a cluster node has failed, or when a network failure means that one or more nodes must be excluded from accessing shared file systems, the remaining nodes perform a rapid recovery process, coordinated by the Storage Control Layer. This involves several steps:

1. The Storage Control Layer ensures that the failed servers no longer have physical access to the shared file system: after the cluster has concluded that servers should be excluded, those nodes must reestablish contact with the cluster and negotiate an orderly re-admittance.
2. The DLM then performs a recovery process, which ensures that locks held by the failed nodes will be released, while briefly pausing new lock grants.
3. While new lock grants are paused, the remaining servers consult the shared file system journal to recover any in-flight transactions underway from the failed servers.
4. At this point, the DLM opens up the file system to new lock requests, and operations can proceed as normal.

It is important to point out that the recovery process does **not** require that surviving nodes drop the locks that they hold, or flush the potentially large amounts of data that they have already cached. This is one of the key reasons that the Clustered File System can provide very fast failover.

Administrative logs and diagnosis tools

The mechanisms previously described together provide the cluster with comprehensive protection against various forms of failure. Almost as important as ensuring that a cluster can survive a failure while preserving data integrity and application availability, however, is enabling administrators to diagnose and remedy problems easily when they do occur. In a complicated system, where a failure of one component (a network link, for instance) may have cascading effects (the isolation of a server from the cluster, and the migration of its applications to other nodes), it is vital to have appropriate information and tools readily available to the operator. CFS's design reflects this need.

CFS maintains administrative logs of cluster activity to aid in management and diagnosis. All significant cluster events are recorded: node reboots, administrative addition or removal of nodes, failure detection events, application failover operations, and so on. The cluster file system allows all logs, including logs of per-server activity, to be available at all times, even if a particular node is down.

In addition, CFS was designed to ease administrators' tasks and simplify diagnosis. The single view of cluster status and ability to manage all of the CFS configuration information from any node dramatically simplifies the task of running multiple machines. Furthermore, when a failure occurs, the administrative interface provides a highlighted list of logged events; the administrator can click each event to see the current status of the effected component. Even in large and complicated cluster configurations, this makes it easy to drill down to the relevant elements. For further investigation, the interface can filter historical log events to show just those relevant for a particular cluster element and can be configured to send notification messages to pagers when particular events occur.

Multi-path I/O

CFS integrates support for multi-path I/O, providing redundancy in cluster nodes' connections to storage. Servers in the cluster may have multiple connections to the SAN; CFS will monitor the health of these connections and move traffic appropriately when a failure is detected. In addition, CFS supports fabrics with multiple switches and multiple attachment points from a storage array to the SAN, allowing configurations to be built in which the cluster can transparently survive the failure of a Fibre Channel interface on a server or on an array, the failure of a Fibre Channel switch, or the disconnection of a cable.

Clustered File System summary

The HP StorageWorks Clustered File System was designed from the outset to meet the requirements of the data center.

Its symmetric design removes performance bottlenecks and allows superior scalability even for write-intensive, commercial workloads.

- It has a comprehensive recoverability and availability architecture, providing a high degree of resilience in the face of hardware failures. This includes extensive support for server and network health monitoring, and support for techniques like automatic multi-pathing of cluster communications to enhance reliability.
- It also provides extensive support for application availability.
- It includes industry-leading support for controlling clustered access to shared storage resources, guaranteeing data integrity even in split-brain scenarios, and protecting administrators from easy-to-make mistakes.
- Finally, in the event of trouble, it provides extensive tools to allow an administrator easily to diagnose and fix the problem.

Other cluster file systems

Red Hat GFS

Red Hat GFS originated in a numerical computing environment, supporting scientific workloads characterized by large sequential reads and writes. Initially, recovery required the cluster to be taken down, and caching required specific support in the storage hardware for a SCSI command extension. Later, online recovery was added. The current version uses a centralized lock management node that handles all lock requests for all nodes in the cluster, but, because the locking scheme is inherited from the SCSIextension-based approach, GFS uses a write-back cache. That is, all updates are immediately written through to disk, which undermines caching benefits for random write workloads. In terms of the taxonomy previously described, GFS started as a scientific computing file system, and has been moving to incorporate elements of a master server–based design.

VERITAS SANpoint Foundation Suite HA

VERITAS SANpoint Foundation Suite HA (SPFS HA) is a master server–based design. SPFS HA uses a master server for metadata operations, which is a scalability limitation. It does, though, employ a symmetric lock manager (DLM) for managing cache coherency. The VERITAS DLM, however, uses a protocol based on global atomic broadcasts for lock acquisition, which causes rapid growth of network traffic as nodes are added to the cluster and therefore poses a scalability limitation.

IBM GPFS

GPFS targets numerical computing workloads. In the past it has been used on IBM's RS6000 SP2 supercomputers to support large-scale scientific computation. Now available on Linux, GPFS uses an in-kernel module to forward I/O requests to a set of user-level processes, which perform file and data operations on behalf of clients and coordinate cluster operations. GPFS can use either IBM SAN infrastructure for shared storage, or can support spreading data across direct attached storage on each cluster member node, with nodes performing data movement over the cluster interconnect. GPFS's design is thus best adapted for numerical computations with predictable, largely sequential read and write requests.

Conclusion

Previous efforts at designing cluster file systems have fallen short of the requirements necessary for adoption in the data center. At base, these file systems have either targeted dramatically different workloads (scientific computing), or, if targeted at commercial workloads, they have opted for an architecture, using a master server to perform cluster-wide operations, which fundamentally limits scalability. By building a **robust, fully symmetric** cluster file system from the ground up with shared storage fabrics in mind, HP has avoided the difficult problems that limited previous approaches.

For more information

For more information on HP StorageWorks Enterprise File Services Clustered Gateway, visit:

http://www.hp.com/go/efs

4AA0-0899ENW, 06/2005