Modern File Systems and Storage

Compiled by Rodney R. Ramdas
rramdas@qualogy.com

Competeta IT B.V
&
Qualogy Consultancy B.V.

Abstract

In this time of Fibre Channel and Gigabit Ethernet, network enabled storage is leaping into new directions. This Whitepaper explores some of the advances in filesystems and storage technology that will propel high-performance computing into the next millennium and aims at being a field guide for enterprise storage managers.

There is no question as to the importance of a file system. Operating systems must provide users with facilities for persistent storage and just as importantly management of data. Filesystems do just that; they allow users to organise, manipulate and access data usually presented to them through a container of data called a file.

According to a recent IDC report Unix system managers added 110% more storage capacity in 1998 than in 1997. System managers interviewed by IDC report specific system capacity is doubling in 12 to 36 months.

This is not surprising because the price-per-megabyte of disk-based storage systems is declining 40% per year. The ability of disk drive suppliers to double the storage capacity every 18 months while increasing performance and keeping prices nearly constant is fuelling the capacity growth. A tenfold price-per-megabyte decline since 1993 is opening new uses and demands on disk storage, its deployment, and IT management.

"Storage must be viewed as a system as a whole that if managed properly delivers services, protects your data assets. Proper management should provide high availability access, performance, complete data security and provide for storage growth at a reasonable cost. [FC Assoc.)"

Storage is more than just a collection of disks and tapes. Storage is something users want available at all times as an unlimited resource.

Overview of existing file systems

Local Filesystems

Local filesystems manage data stored on disks connected directly to a host system. In this approach the user communicates through the I/O subsystem with the underlying filesystem to process requests to open, create, read, write and close files on disk. In order to enhance the users interface to the disk normally a storage abstraction is presented in numerous ways, for instance a logical disk could be a portion of a disk referred to as a partition or these partitions could be residing across multiple disks referred to as a logical volume. The latter case can provide data mirroring (redundant copying of all data), striping across multiple disks (increased throughput) or advanced features such as the ability to dynamically grow logical volumes.

It is important to realise that logical disks or volumes are storage abstractions . To the filesystem itself a disk is a linear sequence of fixed-size, randomly-accessible blocks of storage.

Traditionally filesystems provide a single, persistent namespace for each disk or logical volume by creating a mapping between the block found on disk and the files and directories found on the disk. Since these are attached locally to the host, there is no need for device sharing semantics to maintain the persistent namespace image. Instead aggressive caching and packing filesystem operations are deployed in order to limit the number of disk accesses to provide enhanced performance.

Local filesystems are the building blocks of storage solutions. Two modern solutions are offered by SGI (XFS) and Veritas (VxFS).

Apart from the obvious features required (read, write, open, close etc.), today's enterprise computing relies on advanced features such as extent-based allocation and journalling. When shopping for some medium one can entrust valuable data to, three principles should be taken into account:

- preservation of data integrity,
- availability,
- performance.

Benchmarking has shown extent-based, journalled filesystems to provide the highest amount of integrity, availability and performance.

Both XFS and VxFS provide journalling, a technique whereby system writes are first submitted into a sequential log file which enables fast disk writes. In case of a disk failure the log kept can then be replayed to reconstruct data and guarantee data integrity. VxFS provides this by issuing a circular intent log whereby all filesystem changes are written into this log and periodically flushed to their actual corresponding disk blocks.

Instead of the traditional way of maintaining linked lists of fixed-size blocks with direct, indirect, double indirect or even triple indirect blocks such as in UFS and the Berkeley Fast Filesystem (FFS), in a block map both VxFS and XFS use extent maps to manage disk blocks allocated to a file. An extent map is a starting block address and a length (expressed as the number of blocks). When adding storage to an extent-based filesystem, instead of allocating a block at a time an extent is allocated. This allows for multiple block disk I/O (contiguous mapping) which is considerably faster than the traditional block approach.

It is a well known fact that contiguous I/O provides the highest amount of performance from a filesystem. This is where extents as opposed to simple lists excel. An extent is able to potentially map a much larger area of disk space contiguously. For example if a filesystem used 8-byte block addresses, suppose an extent has length field of 2-bytes, allowing the extent to map up 65536 contiguous file system blocks. Oppose this to a 1024-byte block file system with 8-byte block addresses using a double indirect block. In this case 128 indirect blocks can be referenced that each can then reference yet another 128 data blocks which amounts to 16384 data blocks that can be mapped. If one takes into consideration that extents can be of variable size you end up with a very powerful and high-performance filesystem.

XFS has the curious but effective way of managing disk blocks by utilising two complementary mechanisms: a B+Tree (indexing method) for sorting free blocks by starting block number and another B+Tree for sorting free blocks by their length. This amounts to a filesystem that is both fast for extremely large files as well as for a directory with many small files. XFS assigns small areas on the disk to write indices for the location of data in large files. Generally these indices point to the extents containing the data. The use of B-Tree indices increases performance by avoiding slower multi-level indirect searches of the filesystem data structures – especially when accessing blocks at the end of large files. Most symbolic links and directory files are small files. XFS allows these files to be stored in inodes for increased performance. XFS also uses delayed writes to wait to gather the entire small file in the buffer cache before writing to disk. This reduces the number of writes to disk and extents used for a file. [Holton, Das].

VxFS version 4 can now support files up to 8000 terabytes in size as opposed to XFS's 1 terabyte. SGI has promised up to 9 terabyte files in the near future. However with the recent Open Sourcing of the XFS code this might take a while longer than was expected. Veritas has a couple of rather unique features over XFS such as Quick I/O, which allows applications to access preallocated VxFS files as raw character devices. This greatly enhances performance, particularly in database environments. With regard to NFS which is discussed later, Veritas also offers the possibility of storing the VxFS intent log on a separate physical volume, the so-called Accelator device which greatly enhances NFS performance. One last critical feature is support for Direct I/O whereby data is no longer copied between user space and kernel buffers but rather in and out of user space buffers immediately. Combined with large extents this yields near raw disk performance.

Local filesystems such as the above are almost always used in enterprise computing environments as building blocks for networked and distributed filesystems.

Network Filesystems

Network Filesystems extend the paradigm laid out by local filesystems to include device sharing to users across a network. The user view of the filesystem is that a remote filesystem on some host appears to be locally mounted. To achieve this two prerequisites are necessary: a client-side component to intercept filesystem calls to access files stored on some host and a server-side component that actually hosts the disk that is being shared across a network.
Typically the server has a means to interface with the remote client using a well-defined protocol (e.g. UDP or TCP) and secondly it interfaces with the local filesystems to obtain data for the requesting client. In this scheme the client-side component and thus the user is always aware of data residing on some server. The namespace provided to the user can not easily be made into a single, persistent one with out resorting to extensive network client-server software.

A well known networked file system is the NFS file system introduced by Sun Microsystems in 1985. NFS has since become the de facto industry standard supported by virtually all UNIX's and several non Unix platforms. Independent market research forecasts have shown strong growth in NFS systems from 8.5 million nodes in 1994 to 12 million in 1997.
In addition to developing NFS Sun also developed the so called vnode/vfs installable file system interface that allows a single UNIX kernel to easily support multiple filesystems. The vnode/vfs interface provides for a mapping between filesystem dependent and filesystem independent portions of the kernel. The technique described by the vnode/vfs interface has become the standard in today's modern operating systems. In the case of NFS, clients requests are redirected through the vfs layer onto the network via RPC, XDR and UDP or TCP protocols to the servers vfs layer. This implies NFS is a stateless protocol. The server than translates the requests to the local filesystem.
One of the more pressing disadvantages of a stateless system like this is in that clients cannot proceed writing data until the NFS server has written data to the disk medium. NFS3 relaxes the requirement of writing to stable storage before requests completion. In NFS2 and NFS3 clients and servers cache data in system memory to improve read performance. NFS3 also caches writes. File data consistency is acquired by clients verifying file modification times with the servers.
A frequently overlooked issue with NFS is that NFS does not offer management of data as for instance AFS or Coda do (discussed below). NFS servers simply use local filesystems to store data. However NFS's statelessness helps provide simple crash recovery. If an NFS server crashes, it may begin accepting client request after reboot as if nothing happened. This is similar to the client case: since the server doesn't know anything about clients it is not affected if a client drops off the network. The unfortunate corollary is that in the case of a heavily-loaded NFS server, clients may continue to send requests effectively flooding the network and/or (over)loading the server.

NFS is very portable and provides a high degree of connectivity however the design fundamentally lacks certain desirable features. Single server systems may become bottlenecks as the number and size of client requests increase. NFS version 4 is currently an Internet-Draft counting well over a hundred pages. It sets out to address some of the issues above as well as provide strong security (Kerberos V5).

An interesting offspring of NFS are dedicated NFS Servers. Usually vendors package servers with network adapters, disks and memory tuned to provide customers with an NFS solution. However some vendors, such as Auspex and Network Appliances, provide dedicated, architecturally redesigned NFS servers. In the Auspex Functional Multiprocessor Architecture (FMP), Auspex has recognised that NFS systems contain two major subsystems: the network and the filesystem. This translates into the core building blocks of their systems - the I/O Node. The I/O Node contains two processors that are used asymmetrically in order to provide two services: one processor referred to as the NP (Network Processor) is used to process network protocols and manage caching, the other processor referred to as the FSP (Filesystem and Storage Processor) is dedicated to managing filesystems and storage hardware.

Distributed File Systems

Distributed File Systems try to completely hide the underlying physical location of data to the user of the filesystem. In other words, the filesystem provides a single, persistent logical view of the namespace the user moves in. This means a single pathname to identify a file is all that is required. The user does not need to know or be exposed to the physical location of the file (location transparency). In order to provide this functionality distributed file systems typically provide a client-side component and a server-side component just like network filesystems, however the view offered to the user is managed by special software that network file systems lack i.e. distributed filesystems often incorporate the basis of system management. The software implements a single virtual root directory onto which the entire file hierarchy is mounted.

An example of a distributed file system is AFS or DCE/DFS, both provided by IBM/Transarc. AFS (and DFS, a port of AFS for the DCE environment) both provide a single uniform namespace that is independent of storage location. AFS has distinct clients and servers. Data is stored centrally in a collection of trusted AFS server called "Vice" which are surrounded by untrusted AFS clients (AFS incorporates Kerberos 4 mutual authentication). A client workstation can access any file in the AFS namespace using the same name thus providing location transparency (NFS file names may be different on each client depending on where the exported directory is mounted on each client). It organises files in logical units called volumes. Volumes are quite different from partitions. One advantage is that volumes can be moved around at will from one location to another without affecting users, which is useful for capacity and load balancing. AFS also provides a storage solution across a WAN or the Internet by defining cells effectively creating a world-wide interconnected filesystem which provides in the need of a multinational company requiring a single consistent view from anywhere in the world. AFS sports an aggressive caching principle. Before AFS3 clients would cache recently accessed files in their entirety locally. Since AFS3 files are split up in chunks and cached accordingly. Client cache management is done by the servers who notify clients whenever cached data has become invalid. This has an added benefit of removing the overhead of name lookups from the server to the clients since clients cache entire directories and parse the filenames themselves.
AFS being a stateful model adds difficulties of its own however. As noted AFS caches data on client machines to reduce requests directed at file servers, thereby reducing network and server loads. However on the client side things are less impressive. The combination of client-side caching and callbacks (AFS servers promise AFS clients it will notify them of changes to files) can lead to race conditions and potential deadlock situations in particular when performing read and write operations on very large files. On low end desktop machines client performance can be less than satisfactory since the process of accessing and caching data can be resource and time-consuming. The recent releases of AFS 3.5 address these problems.

A sibling of AFS is the Coda filesystem. Its implementation details are very similar to that of AFS. Coda is a file system designed for large-scale distributed computing. It provides resiliency to server and network failures through the use of two distinct but complementary mechanisms. One mechanism, server replication, involves storing copies of a file at multiple servers (also provided by AFS for read-only data). The other mechanism, disconnected operation, is a mode of execution in which a caching site temporarily assumes the role of a replication site. Disconnected operation is particularly useful for supporting portable workstations. The design of Coda optimises for availability and performance, but provides the highest degree of consistency attainable in the light of those objectives.

On the border line between a later topic in this paper (SANS) and distributed filesystems, lies an interesting initiative by Berkeley - the xFS filesystem (not to be confused with SGI's XFS). While local and distributed filesystems still rely on a central server machine, xFS introduces a serverless system that utilises co-operating peers (workstations) for all file system services. Any machine in the system can store, cache or control any block of data. The net result is redundant data storage since any similar component in the entire system can assume responsibilities for one particular failed component. The xFS developers have recognised that in traditional filesystems the central server is a performance and reliability bottleneck. It is interesting to note that the xFS filesystem has been fuelled by the introduction of switched local area networks such as ATM. Switched networks enable the serverlessness by providing aggregate bandwidth that scales with the number of clients on the network. Another enabling technology such the use of ATM in xFS has been Fibre Channel which is detailed later in this paper.

Criteria for Evaluation

Centralised file systems allow multiple users on a single system to store files locally. Networked and distributed filesystems extend local filesystems by allowing users to share files across different machines interconnected by some communications network. These networked and distributed filesystems are dependent on the well-know client-server concept. In order to evaluate which of these filesystems are important in any particular environment the following checklist has been defined [Vahalia]:

- **Network transparency**

Clients should be able to access remote files using operations that apply to local files.

- **Location transparency**

The name of the file should not reveal its location on the network.

- **Location independence**

The name of the file should not change when its physical location changes.

- **User mobility**

Users should be able to access shared files from any node in the network.

- **Fault tolerance**

The system should continue to function after failure of a single component (a server or network segment).

- **Scalability**

The system should scale well as its loads increases. Also, it should be possible to grow the filesystem incrementally by adding components.

- **File Mobility**

It should be possible to move files from one physical location to another running system.

Moving On

In each of the above systems, storage devices are always attached directly to a server which poses problems in itself (central control means a single point of failure) . The next step would be to disconnect the storage from the servers and connect them instead to their own network commonly referred to as a fabric. This leads us to Storage Area Networks and Network Attached Storage.

Storage Area Networks (SANs)

The Fibre Channel Connection

In 1993 Hewlett-Packard, IBM and Sun Microsystems formed the Fibre Channel Systems Initiative (FCSI) to develop an industry wide standard I/O interconnection that would improve I/O performance an configuration flexibility. The result of this is Fibre Channel (FC), a standard for connecting electronic equipment. In Fibre Channel jargon, the communication network is called a fabric and each piece of equipment connected to the network is called a node. [Adaptec].

Fibre Channel was introduced to provide the following [Tang]:

- Bandwidth to service clients and maintain data availability: Current FC-AL (Fibre Channel Arbitrate Loop) standard provides 1 Gigabaud with planned enhancements providing 2-4 Gigabaud.

- Scalability for long term rapid growth: SCSI is limited to 7 to 15 storage devices. FC-AL provides 126 nodes. In addition FC-AL cabling permits up to 10 km of length. Furthermore more loops can be added offered virtually unlimited scalability.

- Flexibility to provide optimum balance of server and storage capacity: FC-AL provides unique possibilities though the use of modular networking devices such as hubs and routers enabling enhanced availability and advanced load balancing.

- Manageability for ease of installation: The following open standards for SAN management apply:

- SCSI commands set
- SCSI enclosure Services
- SCSI self monitoring analysis and reporting technology
- Simple Network Management Protocol (SNMP)
- Web based enterprise management (WBEM)

The advancements made in Fibre Channel have been the fundamental driver for the development of so called Storage Area Networks. The FCSI design goal was to define a universal physical communication medium that could support multiple functional protocols simultaneously. Currently SCSI and TCP/IP are at the advanced stages of standardisation.
FC provides system architects with three basic topologies:

1. Point to Point
2. Fabric (switch)
3. Arbitrated loop (ring)

A FC network can be set-up as following: a single disk attached to a single computer uses a point-to-point connection, a group of disks attached to a single computer would use an arbitrated loop to reduce per port cost per disk and a cluster of large servers might share several fast disk arrays across a Fabric switch. This allows for bandwidth and interconnect capabilities as demanded. This ability brings new levels of capability and performance to existing network infrastructures. For instance the FC standard includes copper cable, fibre channel disks, fibre optic cable, gigabit link modules host bus adapters, hubs, stackers ,extenders, routers etc. These are building blocks advanced sites have readily available today. According to the Gartner Group by the year 2000 70 % of shared storage will return to centralised management. This works in favour of Fibre Channel solutions since FC provides the storage manager with 2 significant advantages:

1. Fibre Channel networks provide integrated storage with the performance and efficiency of a distributed architecture and
2. Fibre Channel at the same time provides the reliability and manageability of centralised data storage.

Most enterprise sites require a storage management strategy that can provide for end-to-end data management (from online-storage to backup storage). Fibre Channel has an inherent capacity to meet these business requirements by implementing the well-known SNMP and SCSI Enclosure Standards for management. The FC catalyst provides the means to move from local, networked or distributed filesystems and storage solutions to Storage Area Networks. A SAN consists of external, centralised data storage connected in some configuration consisting of hubs , switches, routers etc.

A SAN is very similar to a LAN or WAN but is designed and dedicated to handling storage independently of (LAN/WAN) network traffic.

Equally importantly

A SAN still requires a server or cluster of servers running a network or distributed filesystem to transport data from the SAN, onto the LAN/WAN and hence to clients.

Most SANS operate using Fibre Channel arbitrated loop technology. In this way every device connected to the loop has an address (up to 126 nodes can be connected though no more than 50 to 60 nodes is advised).

Fibre Channel is not synonymous with a SAN. It takes four components to set up a SAN.

1. An interface such as Fibre Channel.
2. Interconnects like switches , gateways routers or hubs.
3. Some governing protocol like SCSI or IP that controls the traffic.
4. Nodes consisting of storage devices and servers.

This combination results in a dedicated network from which machines can access and manage shared storage. In the future and to some extent right now a SAN can support increasingly complex server-to-storage functions such as fault-tolerant access paths with automatic failover, the dynamic reallocating of storage devices, assignment of dedicated storage space within a device (logical unit masking) for specific hosts or operating environments, and high-availability clusters. A SAN re-creates a very old concept in computing: *storage-centric management* methodology. Currently advances in this area are being made particularly by Veritas, the Fibre Channel Association, Compaq and Sun Microsystems. A SAN moves away from the paradigms described earlier, i.e. from a one-to-one (local storage) or few-to-many (network/distributed storage) to a many-to-many approach. A SAN is a separate network with fibre as its transmission medium. The fabric can contain anything from hubs, routers , switches etc. which connect to a set of heterogeneous servers. It also exploits the benefits of Network Attached Devices which connect multiple disks and tapeunits directly to the network that can then be shared among multiple servers. The main motive to move enterprise storage onto it's own dedicated fibre network is to provide an I/O path separate from the main LAN. This frees up the LAN to become a cost effective application and data sharing environment while leaving the Fibre Channel SAN to as a low-latency, high bandwidth dedicated I/O path. One Fibre Channel Arbitrated Loop can connect up to 126 nodes consisting of server, workstation, storage systems and tape libraries The best way to look at a SAN is to look at an actual SAN configuration and deployment. An impressive demonstration of the SAN concept was provided by Veritas at the Storage Area Networking event in 1998. During this event a 32-node SUN cluster utilising the VERITAS Cluster Server was demonstrated running within a complete SAN environment comprising Sun servers, fibre cards, switches and a Sun A5000. The demonstration sported three principles:

- ***high availability:*** through the use of clustering software and fibre technology constant availability could be provided

- ***drive sharing :*** with the use of a fibre channel hub a cost effective as well as high performance backup strategy was demonstrated

- ***SAN management ;*** through the use of visualisation software a complete SAN environment was shown

Full details can be found at http://www.veritas.com/newtech/san/events.html.

A few further points regarding SANs:

1. The complexity of designing a SAN is daunting. Proper system design/management and interoperable hardware are needed. In November of last year 3Com, Legato Systems and MTI Technology announced the StorageConnect Compatibility Program which would focus on creating compatible products for use in SANs. However as of March 3Com has withdrawn from the alliance. 3com is abandoning SANs altogether. They believe SANs will make a killing in the industry but not by networking hardware companies. 3Com is dedicated to Gigabit Ethernet not Fibre.

2. IDC forecasts $11.4 billion in sales amounting to 37 percent of all server storage in 2002 will be SANs.

3. Other agreements have been made, the Fibre Alliance consists of EMC, HP, Legato, Veritas, Gadzoox and others.

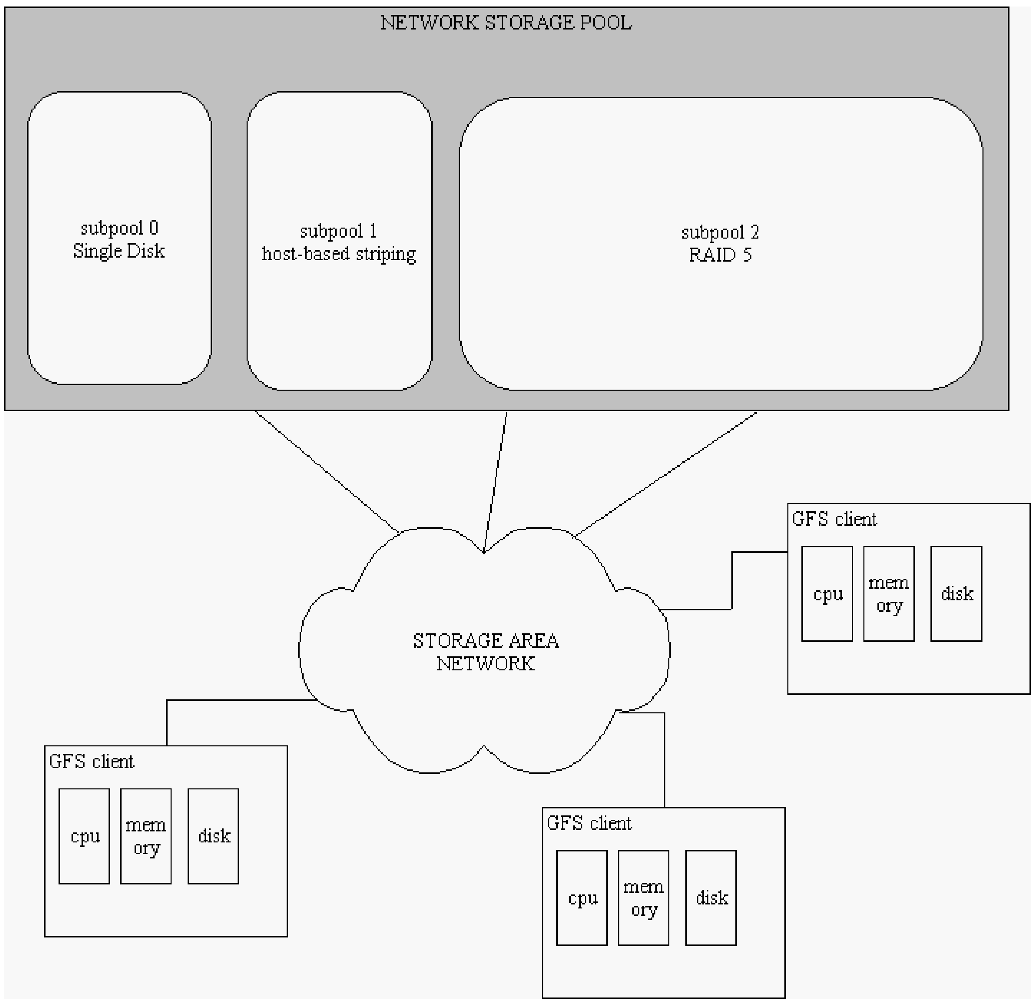
4. Dataquest Tom Lahive: Lahive thinks SAN technology will progress in three distinct phases. Phase one will bring compatibility with industry standard host adapters controllers, tape libraries, and alternative brand switches and hubs. General availability of this hardware will come in the second or third quarters of this year, Lahive says. Phase two will include node path allocation, the ability of storage devices to intelligently manage paths to and from the switch/hub in order to increase availability and performance General market availability and acceptance will start later this year and will continue through the first half of 2000. In phase three, Lahive believes storage management utilities will be hosted on a dedicated system and agents will be embedded on storage network devices and storage systems. Some basic system resource management functionality is available now, though robust core and backup functionality will not receive general market acceptance until 2000.

Back to Filesystems

Network attached storage and Storage Area Networks pose interesting problems for filesystem developers. Shared filesystems that exploit network attached storage are not new. Examples of this kind of technology are Cray SFS, DEC Vaxcluster and Oracle Parallel Server. All of these provide serverless distributed filesystems. However they rely on proprietary network-attached storage or expensive custom interfaces such as HiPPI. What is needed for true Network Attached Storage is a serverless filesystem that fully exploits new interfaces such as Fibre Channel. This approach would allow all client machines to have full access to all storage devices on the network, making access more efficient and reliable. Scaleable networked storage interfaces like Fibre Channel will allow computer architects to design systems with many shared storage devices, increasing the performance and reliability of the design.

The Next-Generation

One such solution is the Global File System (GFS) file system developed by The GFS Group and sponsored by such companies as Brocade Communication Systems, NASA, Seagate Technology and Veritas. GFS was originally designed and implemented on SGI hardware exploiting Fibre Channel technology to postprocess large scientific datasets. The GFS group eventually decided to port their IRIX code to open source Linux for implementation reasons (IRIX is closed source making it difficult to integrate GFS into kernel). One distinction made by [Soltis, Ruwart, O'Keefe] in distributed filesystems is on the one side message-based sharing and on the other shared storage. In the former data is shared by communication between machines across a network with the data stored locally on device within each machine. An example of this type are NFS (RPC on top of UDP or TCP datagrams) and AFS/Coda (RX on top of UDP datagrams). The strength of these architectures lies in their extensibility. They are as extensible as the network protocol is that they use. This might explain the domination on NFS over AFS in the industry. However message-based architectures are hard to load-level since it depends on dynamic properties of virtually all nodes in their network e.g. type of data (attribute or data intensive) and machine capacity. In badly managed message-based file systems/networks the investment in high speed devices like disk arrays is virtually negated by bandwidth limitations of the network, which is the bottleneck all too often. As noted before advances in Fibre Channel but also internal architectural advancements such as the Gigaplane-XB[SUN] interconnect in Sun Microsystems's E10000 (Starfire) address these problems. Shared storage as implemented in GFS offers uniform data access to all devices on the network. Machines and storage devices are connected in a Fibre Channel network (the fabric). In this way all storage is assembled as it were in a Network Storage Pool (NSP) – a collection of network attached storage devices logically grouped to provide node machines with a unified storage space. The pool is not owned or controlled by any one machine but rather act as a shared storage to all machines and devices on the network. GFS logically groups storage devices in subpools that represent similar device types. The crux of the GFS file system is what is known as fine-grained locking principle which the GFS group developed for the SCSI-3 command set in association with Seagate who have actually implemented the principle in their high-end disks. Since a design principle of GFS and shared filesystems in general is to allow multiple clients to access and manipulate shared storage devices simultaneously a lock mechanism is required to implement a mutual exclusion mechanism.



The Network Storage Pool illustrated above is accessed by a unique storage address which is provided by a device driver layered on top of SCSI and Fibre Channel drivers. This NSP driver translates the logical address space of the file system to the actual address of each device. GFS specifies file systems into several resource (mini-filesystems) groups that are then distributed across the entire NSP. A resource group consists of information similar to traditional superblocks like data bitmaps and datablocks. The actual GFS superblock is available in-core on each client and contains information about mount permissions and device names. Inodes are based on the actual disk addresses. Locking can be performed in either clients or in the actual devices. Locking on clients implies sophisticated and therefor complicated caching and sharing mechanisms which is the reason why the GFS group has opted for device-level locking. The net result of the locking mechanism is quite similar to the way shared multiprocessor computers access memory, by utilising this principle GFS is able to provide transparent parallel access to storage devices while maintaining UNIX file system semantics.

Conclusion

The filesystem and storage technology industry seems to be definitely moving towards Fibre Channel and Gigabit Ethernet connected storage devices. This might require rethinking of various filesystem and storage paradigms that we have come to cherish over time. However it is important to realise Copernican Turns don't come easily in an industry where high-availability, fault-tolerance and data consistency while maintaining 24-7 full service availability is of the utmost importance to mission-critical enterprise computing.

References

1. Competa IT bv: http://www.competa.com (Unix Architecture)
2. Qualogy Consultancy B.V. http://www.qualogy.com (Oracle Architecture)
3. Sun MicroSystem's, "GigaPlane InterConnect": http://www.sun.com/servers/highend/10000/Tour/interconnect.html
4. "The Global File System" [Steven R. Soltis, Thomas M. Ruwart, Matthew T. O'Keefe] Appeared in the Proceedings of the Fifth NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies, September 17-19, 1996, College Park, MD
5. Shared File Systems and Fibre Channel , Matthew T. O'Keefe Appeared in the Proceedings of the Sixth NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies, March 23-26, 1998, College Park, MD.
6. 5. "Network-Attached Storage , A Compelling Story for Storage Consolidation" An IDC White Paper by analyst Robert C. Gray
7. "Auspex 4Front System Architecture", Auspex Engineering Technical Report 24, January 1999
8. The Storage Networking Industry Association http://www.snia.org/
9. The Fibre Channel Community http://www.fcloop.org
10. Berkeley NOW Research http://now.cs.berkeley.edu/nowResearch.html
11. "Unix Internals, The New Frontier"' by [Uresh Vahalia] ISBN 0-13-101908-2 Prentice Hall
12. The Fibre Channel Association http://www.fibrechannel.com
13. "Fibre Channel, Storage Area Networks, and Disk Array Systems", Adaptec http://www.adaptec.com
14. Veritas http://www.veritas.com
15. Veritas Filesystem Whitepapers 1-3 available from Veritas website
16. 15. "XFS: A Next Generation Journalled 64-Bit Filesystem With Guaranteed Rate I/0", Mike Holton, Raj Das, SGI, Inc.
17. "Storage Area Networking", Dave Tang http://www.gadzoox.com