

# File System Technology – Thinking Outside the Box

Lance Evans

SAM-QFS File System Group

Network Storage Marketing

Sun Microsystems Inc.

## Abstract

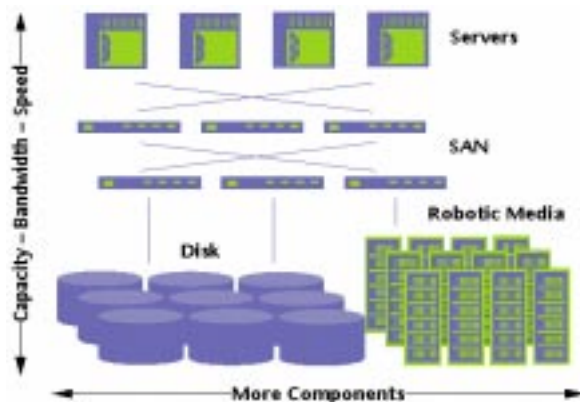
*Storage has historically been tied to a single host or constrained to a single appliance on the network. Differing types of media such as fixed disk, magneto-optical disk, tape, or network attached storage have required different software and access methods to exploit them, constraining them to very narrow ranges of use. Through recent file system software innovation, it is now possible to think outside the single purpose, single-box approach to storage, and to size storage, channel capacity, and server horsepower independently.*

## Introduction

A file's purpose is to ease access to non-volatile storage through a simple, commonly understood interface. Using a file system for access to all system resources is intrinsic to the UNIX® programming paradigm. Yet, storage subsystem complexity, channel connectivity options, redundancy needs, and data sharing requirements strain the limits of traditional file systems. These stresses reduce the viability of the file model when used for the purpose for which the file was invented – storing user data. Though system administrators prefer an easy file system interface, they often must manage raw devices or soft volumes directly and manually target applications to those devices in order to accomplish performance, scalability, or disaster recovery goals. Evolving existing file systems to meet high performance, resiliency, recoverability, and manageability requirements is challenging as file systems scale into millions of files and multiple terabytes of data.

These new demands are being met with novel approaches that have the potential

to release system designers and administrators from constraints thought insurmountable a few short years ago. No longer is there a one-to-one relationship between application, server, channel, and disk. Each element in the data path can be scaled independently to meet performance and scalability objectives without concern for access to data (figure 1).



Clustering numerous parallel nodes has required proprietary high-speed memory interconnects or slow network transports for data sharing. Now the I/O channel can be used instead to efficiently share files among operating system instances. Backup utilities have been loosely coupled to file systems and have required their own external databases for management. Now the file system is capable of backing itself up. File systems have historically been constrained to one hardware device or have been dependent upon underlying volume managers for file system capacities differing from the foundational hardware. Now quotas and advanced file allocation can control where files reside and how much space they can consume, and the file system can grow across large numbers of devices independent of underlying

device aggregation and RAID (Redundant Array of Independent Disks) techniques. And outage recovery has become problematic as file systems grow ever larger, forcing administrators to split, grow, shrink, and reallocate storage amongst numerous small file systems. Self-managing file systems can reduce these administrative headaches and reduce Total Cost of Ownership (TCO).

The author will discuss these issues in detail while explaining the features and uses of SAM-QFS version 4.0 (due 2002 calendar quarter 2), which provides a simple file system interface to a wide variety of transparent, high performance file system services scaling with the largest Solaris™ platforms. Examples and case studies will be provided of practical configurations that leverage the latest technology. And comparisons will be drawn between traditional file management approaches and the advanced file services of SAM-QFS. The audience will be left with practical ways to leverage Sun's new file services beyond traditional single-box approaches.

## File Systems: Benefits and Purposes

A simple file is a named series of bytes that is addressable to byte granularity, and functions as a non-volatile storage location for the computer user's data. UNIX files' name space consists of hierarchically nested directories, with files as leaves within this tree structure. Files can be accessed through command-line utilities or standard library interfaces accessible to application software.

In addition to simple files, UNIX represents many of its other resources as files for ease of access. Everything that is executed on the system is a process, and all process input and output (I/O) is done to a file.<sup>1</sup> Some resources represented as files include hardware components, raw devices, operating system information, communication facilities, links to other files, and the directories that comprise the file system's name space itself. The utility of the file paradigm is evidenced by its continued popularity and ubiquity in modern operating systems – the Solaris Operating Environment is no exception.

When users need to store large amounts of non-volatile data, they typically choose either database management systems or file systems, depending on the desired method of access to the data, performance requirements, the sheer volume of the data, and their need for data manageability. Application developers have the option of writing to raw devices which themselves are represented as files, but the extra application sophistication required to write to raw devices and manage the data thereon, rarely justifies the benefits of marginally better performance and added control over data layout, storage, and retrieval. Database management systems often offer the option of storing data on raw devices, but most administrators still prefer to construct even these higher-level storage abstractions over file systems for ease of data management and backup.

---

<sup>1</sup> **McDougal R, Mauro J: (2001) *Solaris Internals, Core Kernel Architecture*. Saddle River NJ: Prentice Hall.**

File systems provide a beneficial degree of abstraction from the hardware so storage resources can be allocated appropriately to various users. Quota-enabled file systems allow fine-grained command-driven control over the storage space consumed by a given user or community of users, without reconstruction of raw volume groups and without file system rebuilds or risky volume resizing. Access Control Lists (ACLs) provide a flexible means of preventing unauthorized access to data intermixed throughout a global namespace, an impossible task using other strategies that lock a user's data upon a privately owned disk or use simple UNIX mode-based security. For voracious consumers of storage, numerous devices can be aggregated together to present a single file system, and for modest consumers large devices can be carved into small chunks for sharing amongst multiple file systems. With proper file system design, files can exceed the size of a single device, and even provide sparse record storage to meet the most exotic and challenging application storage requirements. All these features are built into Sun SAM-QFS 4.0 and allow a larger pool of storage to be managed more easily and dynamically, reducing or eliminating administrative activities of growing, shrinking, or rebuilding file systems.

Ultimately, the design goal of any file system should be to ease access to and management of large amounts of non-volatile data without reducing performance or restricting the intended use of the system. Indeed, the thirty-year success of UNIX has been attributed in large part to its file system paradigm, which represents an ideal

degree of abstraction from the computer hardware for implementation of most applications. Continued advancements in the file system technology of Sun SAM-QFS are key to this model's continued success.

## File System Architectures Strained

The UNIX file system design predated even the design of the UNIX operating system itself.<sup>2</sup> Various forms of this file system implementation have existed ever since, and many UNIX file systems derive from this common heritage. However, evolution of sophisticated computing platforms, coupled with unfathomed uses and reduced component cost, has driven demand for file services beyond the capabilities of traditional UNIX file systems.

Customers demand file access and sharing capabilities that truly scale in terms of performance, capacity, and utility. Customer demand is growing for multi-gigabyte or terabyte files, and for file systems of tens or hundreds of terabytes. Numbers of files in users' file systems are growing by millions. However, customers continue to expect resiliency and management ease similar to that achievable when file systems were small. But existing architectures become strained because they are often derived from the original UNIX file system design and inherit an outdated set of requirements and implementation details. Most Solaris file systems can be built on only one device, for example,

and Solaris devices are limited today to one terabyte.

Volume management software has arisen to overcome limitations in file and operating system design, aggregating disks together into single larger volumes upon which the single-device file system can be placed. Software volume managers can provide concatenation, striping, and RAID functionality traditional file systems do not, and can flexibly allocate storage to multiple file systems. But volume managers abstract the file system from the devices upon which the data resides. The file system must blindly read from and write bytes into a pseudo-device without regard for the underlying hardware geometry or its performance characteristics. This interaction between two separate software components in the data path increases code path length, increases processing complexity, and negatively impacts scalable performance.

Further, software volume manager architectures introduce an additional layer of indirection between the application and the actual storage devices. This reduces resiliency and increases configuration complexity due to the need to maintain and manage coordination between two independent, yet interdependent sets of references to each file's blocks on disk. Both devices must operate flawlessly in tandem to dereference and access the user's data. This translates to additional processing steps and reduced resiliency.

Changing the size of the single-disk file system requires interaction between the volume manager and the file system to

---

<sup>2</sup> **Richie D: (1984) The Evolution of the UNIX Time-Sharing System.** *AT&T Bell Laboratories Technical Journal*, 63 No. 6 Part 2, 1577-93.

assure consistency is maintained throughout resizing operations. Sharing file systems between servers also becomes problematic because coordination must be reliably maintained between low-level volume manager instances across hosts, and high-level file system instances as well, in addition to intra-host interaction between file systems and their respective volume managers.

Latest generation file systems such as Sun SAM-QFS are architected to minimize this complexity, incorporating the best features of both the volume manager and the legacy file system into one tight code base that is aware of underlying device geometry, spans multiple devices for scalability, and possesses a coherent processing model for shortest code paths and best performance.

## **Overcoming Network File System (NFS) Limits**

NFS has served the UNIX community well, introducing for the first time, transparent sharing of files across sockets-based links. NFS has clearly been the best and most popular choice among UNIX platforms for file sharing since it was introduced in 1985. Technical complexity is lower than sharing devices via physical channels across UNIX hosts. And its single-server model minimizes the software challenges in coordinating views of a file system across operating system instances. NFS has continued to evolve and still enjoys broad use and applicability. But Network Attached Storage (NAS) architectures that

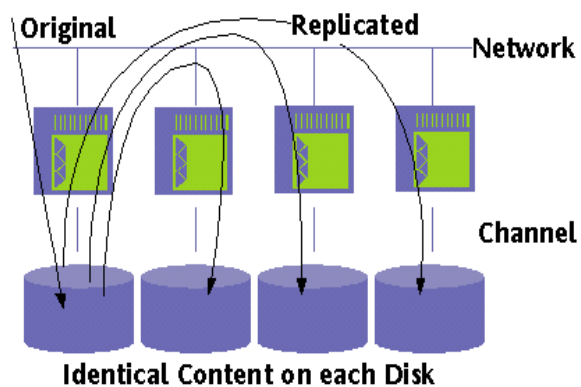
implement NFS or other similar specifications such as CIFS, are limited in two key ways – by the very enablers that make NFS possible. They depend for file data transfer on networks that are neither designed nor optimized for channel I/O, and they must pass the data through a central file server which holds captive the raw storage devices.

Historically, parallel channel-based architectures such as SCSI (Small Computer System Interface) were designed to most simply and efficiently establish electrical connectivity between a single host interface and a few storage devices. Distances were assumed to be short – just a few meters – enabling speedy parallel data movement and low I/O transaction latency. Recent serial channel architectures such as Fibre Channel endeavor to surpass latency and performance capabilities of parallel channels, with greater sharing flexibility than their copper counterparts and better grades of service than networks typical of NFS file sharing.

Fibre Channel and other serial channel architectures achieve this by delegating much of the transport overhead to the channel hardware to reduce software processing steps, and by increasing transmission frequency to offer bandwidth while minimizing intra-packet receipt latency. But most importantly, these channel-side networking protocols make many optimizations appropriate for small, short-distance channel-side networks that are impossible to implement when designing protocols and solutions for network-side sharing software and global WANs. Channel-side architectures do not enjoy the wide

applicability of transports such as Ethernet or protocols such as TCP/IP, but they provide high efficiency and crisp grades of service (either implicitly or explicitly) between the devices themselves and the hosts to which they attach, while providing shared access to storage over high speed switching equipment.

The second limitation in the NFS architecture (as well as in any proxy-based file system such as Sun's Global File System) is that all data must pass through a single controlling host or NAS file server. The capacity and throughput of the entire configuration are constrained to the performance of that single node. When throughput or I/O operation demands rise, the NFS server host can become burdened with traffic, even when acting as little more than a protocol converter between channel-side disk I/O and packetized network file traffic. Scaling performance of a NAS solution beyond the boundaries of a single server requires replicating the data and serving it up from multiple locations, increasing management complexity and storage costs while decreasing coherency of stored data (figure 2).



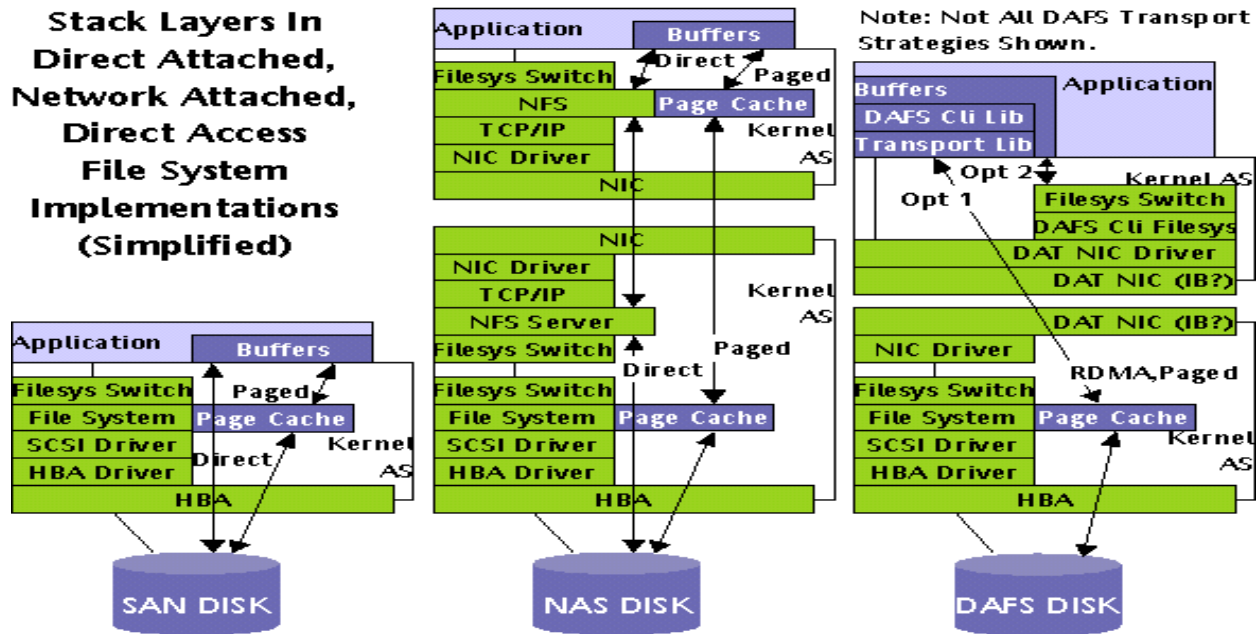
Further, the typical I/O operation latency in accessing files through an NFS client's operating system, a network, and a remotely-attached host (whether embedded in a NAS appliance or not) makes NAS architecturally unsuitable for latency-sensitive applications such as transaction database applications.

Various attempts have been made to compensate for these intrinsic design limitations. For instance, vendors have implemented the entire NFS server in silicon, which incrementally speeds NFS server processing but does not address core architectural constraints.

Others propose implementation of strategies to bypass an NFS client's operating system for increased efficiency. Virtual Interface (VI)/Direct Access File System (DAFS) uses specialized high-speed hardware interconnects (e.g. Infiniband) to provide file services directly from the NAS appliance to applications without client operating system interference. But this approach retains undesirable processing latency on the NAS server at the expense of important client-side operating system services such as caching and error management.

Data normally cached for applications in local file system memory must instead be accessed across a link, increasing bandwidth requirements. To overcome this, additional caching functionality must be implemented by application developers and run in user address space, or implemented in faster kernel memory. Thus the file system must gradually be re-implemented on the client platform, negating benefits of

## Stack Layers In Direct Attached, Network Attached, Direct Access File System Implementations (Simplified)



bypass. The high speed interconnects become just another transport mechanism competing with SCSI, Fibre Channel, and Infiniband NAS solutions. Unproven bypass strategies therefore show questionable benefits in the general case, while being less convenient than benefits the traditional UNIX file paradigm.<sup>3</sup>

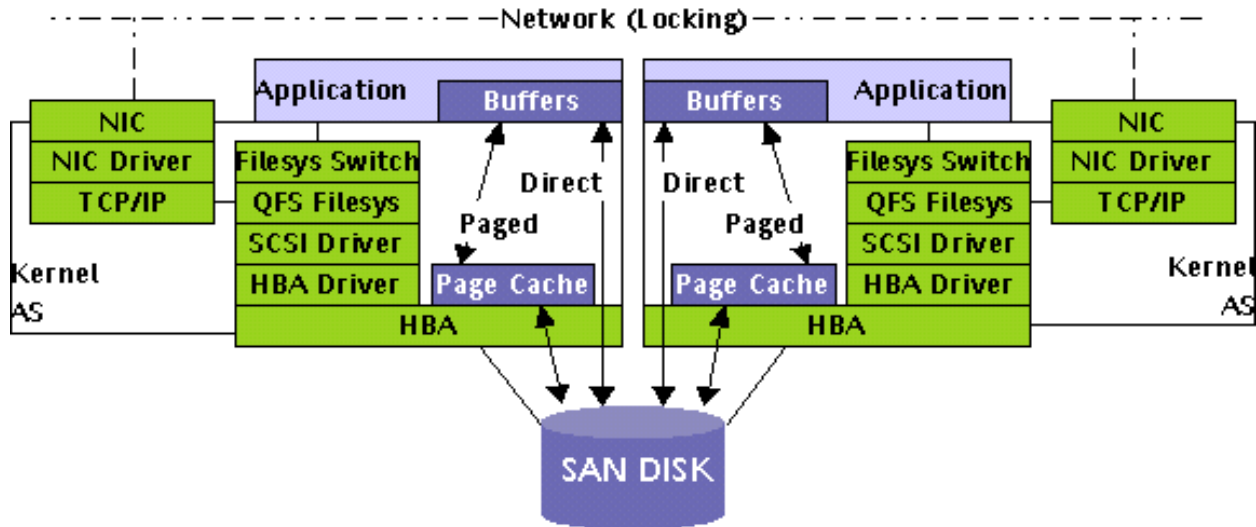
NFS was revolutionary technology when introduced and will continue to evolve. NFS is often the right choice for many distributed applications. However, for customers who need file sharing with high transaction rates and throughput, or who cannot build NFS servers big enough to satisfy customer requirements, the advanced Storage Area Networked (SAN) file system technology of Sun SAM-QFS releases UNIX systems from the constraints of NAS and other proxy-based file sharing technologies.

With Sun QFS 4.0, the name space can be shared across the TCP/IP network, as well as coordination information and other metadata needed between clients, while all the I/O is conducted on the fibre channel. The file system is implemented as kernel code, so efficiency is maintained and applications work unmodified. File sharing can be accomplished with the benefits of direct attached storage and the flexibility of NAS (figure 4).

## Efficiency of Data Path Elements

Between the application and the physical storage medium there are numerous physical links and processing steps regardless of which configuration is chosen. Many thousands of machine instructions must be executed just within the host, to issue and complete a single channel-side I/O to a storage device. Minimizing the number of required instructions is a key to host I/O efficiency.

<sup>3</sup> DAFS: Direct Access File System Protocol Version 1.00, DAFS Collaborative, September 1, 2001



Most file systems are generalized for use across a broad range of customer requirements, and have been modified substantially to achieve reasonably good performance in most circumstances. File system design efforts always compromise between numerous tradeoffs when adapting existing technology to new file access demands. Tunable parameters are added to file systems to allow more broad use with acceptable performance.

But traditional file systems are constrained by their architectural legacy, and must continue to serve existing customer bases having varying requirements. Modification and improvement of the file system becomes problematic as customers' modes of operation may be impacted by file system modifications. Customer sentiment builds around stability versus evolution. Latest file system software technologies, while less mature, leverage thirty years of requirements gathering and the history of UNIX file system adaptation, without the encumbrances of a long code lineage or reluctance to change. Products can be built from scratch, designed from the

ground up to be more flexible, scalable, and faster than their predecessors.

Regarding performance in the data path, latest generation file systems should be designed to minimize the number of instructions required for each I/O, and to intelligently optimize data layout and I/O patterns on devices for optimal throughput. For some workloads the file system cannot be of benefit, as the I/O bottleneck lies elsewhere outside the file system. But the file system should add almost no overhead and be tunable to either allow highly optimized applications raw speed to devices, or to intercede with caching between applications and the I/O subsystem, to compensate for sub-optimal application designs. Advanced file systems provide carefully designed tuning parameters at numerous key I/O choke points, and automatically adapt to application I/O behavior patterns.

SAM-QFS 4.0 offers countless tunables to adapt the file system to a particular user's environment. Two examples of tunables for disk-based data are automatic direct I/O and qwrite (or mh\_write on shared read/write file

systems). QFS can discover separately for reads and writes, sequential I/O patterns based on the number of ill- and well-formed I/Os of specific sizes performed to a file. The file system can automatically disable the Solaris page cache for reads or writes independently for a given file on the fly. And paging is automatically re-enabled when I/O patterns no longer warrant direct I/O. Qwrite allows disabling of the file system's POSIX write lock mechanism to allow thread-aware applications to do simultaneous reads and writes to a given file. The mh\_write feature extends this capability across multiple sharing hosts.

## File Sharing Techniques

Numerous file sharing and movement techniques such as NFS, discussed previously, exist to satisfy a wide range of customer requirements. Some of these techniques are listed below:

- **Backup and restore** – Data can be copied to backup media and restored to an alternate system independent of the original copies. Such strategies are used for managing disaster recovery sites, disaster recovery rehearsal, and collection of remotely generated data.

- **Local and remote replication** – Software can actively keep multiple copies of files consistent through the network using a “push strategy” where the origin and destination systems have adequate access to network resources, when the amount of data is small enough to move through the network, or the urgency of copying the data through the network justifies the cost.
- **Network-side and Cluster Interconnect-based sharing** – NFS, CIFS, PFS, and other networking mechanisms exist for “pull strategies” of file access and distribution.
- **Channel-side sharing** – QFS and SANergy offer mechanisms for inter-host sharing of files stored across block devices on the SAN. Different sharing strategies are effective for read/write, read mostly, and read only implementations.
- **Snapshot, versioning, and views** – Images of the file system can be created in whole or in part, and through various forms of pointer manipulation more than one representation of common data can be presented to applications for parallel access.

All these strategies are useful in specific customer instances, but require separate skills and analysis. Advanced file systems can provide the benefits of all these technologies with a single architectural model, reducing administration complexity and Total

Cost of Ownership (TCO). The file system becomes a policy-driven, automated, universal byte storage service, transparently copying files where they need to go and permitting local or distributed access through a single, simple, file system interface.

An example of this is using QFS 4.0 in a web services environment typically served by file replication methods or NFS servers. Using QFS Shared Reader technology, data can be deposited on a common disk platform and shared with any number of hosts serving the data to the web. Web servers can be outside the firewall without risk to the intranet, since all data is shared only via the SAN.



The business process of managing this infrastructure is much simpler, and with SAM-QFS 4.0 the disk subsystem can automatically be backed up, and also replicated to remote clusters of web servers without additional software (figure 5).

### Backup Restoration, Scalability, and Business Continuance

Traditional backups become more difficult as file systems scale into

terabytes and millions of files. Full backups consume huge blocks of system time and consume massive resources while they complete. Conducting a ten terabyte full backup within twenty-four hours requires constant streaming of backup media at over 120 MB per second:

$$10\text{TB} * 1024\text{GB}/1\text{TB} * 1024\text{MB}/1\text{GB} / 24\text{Hr}/1\text{Dy} * 3600\text{Sc}/1\text{Hr} = 121\text{MB}/1\text{Sec}$$

The amount of incremental data churn and the frequency of incrementals determines the amount of data that must be backed up during each increment. Typically, even small customers do not have the luxury of running incremental backups more than once daily. All work since the previous evening's incremental backup is therefore lost when outages require subsequent restoration.

And restoration of a ten terabyte file system requires even more resources and time than the backups. More than ten terabytes must be moved when the full and all subsequent incrementals are restored. Restoring the system to operation can keep the systems down for extended periods of time, resulting in huge financial impacts to business-critical applications. Typical steps and processing times, given the same environment as above, might be:

- Restore Disk to Original Size – one day
- Restore ten terabyte full – one day
- Restore ten 500GB incrementals (assumes 5% daily churn over ten days) – half day

Total restore time, assuming all the restoration steps complete flawlessly, is close to three days, assuming hardware is immediately available to rebuild the original disk configuration. Reducing this time to six hours requires quadruple the resources, if the goal can even be accomplished. And disaster recovery rehearsal requires just as many resources as full disaster recovery, so only spot checks are made, if ever made at all.

Many operations cannot tolerate the kind of outages these scenarios afford, and are therefore forced to break file systems into many small independent islands, isolating them from one another and increasing TCO due to the additional administrative overhead of managing them independently. Automating the coordination of these many backups presents challenges of its own regarding transportation of the data, media sharing, device sharing, and backup volume set management among other factors.

Mirroring and snapshot techniques can provide more frequent file system images, but are really designed to guard only against file deletion or other processing anomalies. They must be coordinated with the backup cycle to provide benefits in system or site loss scenarios. Systems are only interrupted momentarily when a mirror is broken or a snapshot is taken, but separate technologies are often required for backup versus snapshot, and often those technologies are constrained to a single storage subsystem or a portion thereof.

Block-level incremental backups reduce the amount of data in backups, as may

snapshot technologies that can be coordinated with them. But each constituent block or group of blocks backed up together requires a separate metadata entry in the backup engine's (typically relational) database. Safeguarding huge amounts of generated backup metadata becomes a backup problem in itself for large file systems. Inconsistencies arising within the backup database engine can render all the backups useless, necessitating distribution of function for fault tolerance and giving rise to another whole data processing infrastructure simply to support backup and restore. Ironically, implementing fault tolerance in this manner actually increases the likelihood that a problem will occur due to the increased number of managed components.

And snapshots and incremental backups alone do not address the challenge of quick restoration.

Incremental improvements to a flawed backup architecture is not a solution. The advanced Sun SAM-QFS 4.0 file system already has within it – as expanded inodes and directory structures – a very specialized database highly optimized for referencing vast amounts of file storage across many storage devices. Integrating backup functionality into the file system itself can yield the ability to store all metadata in one tightly controlled location. The file system, aware of even the most minor file changes, can perform continual incremental backups as file changes occur over time. The criterion for backing up files is no longer the time of day the backup window is open, but the length of time since each individual

file has changed. Files can be backed up when they become stable, or can be backed up repeatedly – soon after every change – depending on the customer's requirements. Consequently, the file system constantly keeps itself backed up, completely eliminating the need for backup windows and extensive peripheral backup infrastructure.

Rule-based backup directives can guide the built-in backup mechanism to perform backup tasks heretofore impossible with traditional mechanisms. Filter rules can select and group files for backup based on many criteria (e.g. size, ownership, naming convention, location, etc.) or be directed to selectively skip some files altogether. Advanced criteria can also be specified for disposition of the backup copies onto temporary or long-term backup media, either fixed or removable, locally or remotely across a network.

Disaster recovery becomes a snap, because the file system can restore itself *while the system is in production*. If the file system suffers an outage, there is no need to restore the disk to its original capacity. The file system can be built on whatever disk remains, and the metadata restored into it for immediate use. Though the disks are empty the file system can be mounted and users can gain access to their data. Since the file system itself is aware of all the backup media, it can restore on demand, the most needed files requested by customers. System administrators may then choose to bring key files (or portions thereof) back to disk to speed operations while awaiting the arrival of replacement disk storage.

This same functionality operates in remote vaulting and in disaster recovery rehearsals as well. Metadata from the file system can be dumped, copied elsewhere, and restored wherever there is access to the removable media comprising the backups. Even a small disk subsystem is capable of accepting the metadata, and spot or comprehensive checks of all the backup media can be conducted automatically with a fraction of the resources required to rehearse a traditional disaster recovery scenario.

The upshot is reduced effort and computing resources required to back up and restore enormous amounts of file system storage. Sun SAM-QFS 4.0 provides all these features in a single, comprehensive, file system services package.

## **File Virtualization and Storage Management**

Attached to all computing systems are numerous peripheral devices – sometimes numbering in the hundreds and spanning a variety of storage types. Users must choose which peripherals to target with various forms of data, through manual administrative configuration and repeated allocation. Numerous factors including these devices' functionality, access time, resiliency, and transfer rate determine the usefulness of a particular storage peripheral type. Several examples of data path elements and their typical uses are:

- **Disks** – aggregated together or sliced apart with volume managers into useful units of storage
- **Tape drives** – accessed and controlled through backup applications
- **Optical disks** – accessed and controlled through backup applications or directly from specialized user applications
- **Robotic media changers** – containing tape or disk drives and their storage media, manipulated by software designed to interact with these robotics
- **Network mount points** – organized for transportation and sharing of data

The typical uses of these peripherals is limited by the software's ability to manage them. Most applications store and retrieve data from disk using the simple and ubiquitous file system interface. Some applications (typically designed for backup) also can manipulate removable media and libraries. These elements must all be consciously selected, allocated, and managed by the system administrator for proper function and optimal utilization. Administration at this level is time consuming and often disruptive to operations, reducing resiliency of the platform.

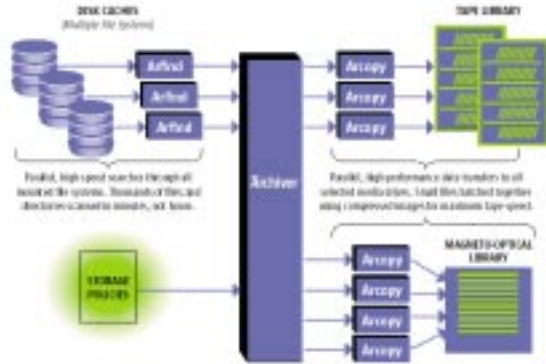
While there is no current method for completely automating all of this activity, an advanced file system can be configured to address all these devices and transparently deposit files in appropriate locations based on user-

defined policies. Manual administration of the underlying devices becomes less frequent and less urgent. Files or portions thereof can be backed up to alternate locations based on customer business policies, and the local disk or cache resources consumed by them can be relinquished for other uses.

Similar to the manner in which the Solaris Operating Environment manages virtual memory, files or portions of files can be shuffled onto or off of the local storage subsystem as they are needed, rather than always needing to be stored on the fastest, most expensive (and sometimes most fragile) media just in case it is needed.

Administrators no longer need to back up and delete old data that is important to retain but is less urgently needed on disk. Retention requirements are no longer met by paper or database records and backup tapes, but by the file system itself. Later needs for the data do not require administrators to temporarily allocate storage and manually restore archived information to disk. All the files appear to be on disk always, and retrieving them is as simple as opening and accessing them. The file system manages capacity on its own, reducing the need for risky and laborious reallocation of storage into and out of existing file systems as their sizes need to be changed. The disk cache is sized for the *expected working set* of files, rather than the required total capacity. Business rules describe the relative importance of user data to the system, and the system manages the data appropriately. And of course, the system administrator can override the

system's automation as needed, to satisfy urgent or unexpected requests.



The advanced storage management capabilities which automate file system backups are also available to virtualize file storage onto whatever media is available. Sun SAM-QFS 4.0 can seamlessly archive-enable applications that have no built-in tape storage or replication functionality. And the whole process can be automated for management of vast quantities of high speed disk, tape, magneto-optical disk, remote vaults, and offsite disaster recovery media.

## Real-World Examples of SAM-QFS

The following section contains abstracts of case studies previously distributed by Sun Microsystems, describing various real-world application of Sun SAM-QFS. For further information on any of these implementations, see your Sun Sales Representative.

### Lotus Notes Archival<sup>4</sup>

With assistance from a Notes programmer from Lotus and a Sun SAM-FS specialists, Sun Microsystems GmbH has assembled an email archival solution based on Sun SAM-QFS.

With minor changes to the mail template and implementation of some additional functions and two agents:

- The user can decide which documents he wants to archive
- Documents will be automatically archived after a period of time (e. g. 30 days)

Archival is conducted at night, causing the documents that have been marked for archival to be transferred to archive storage on a Sun SAM-QFS file system. After the agent is finished, the archived Notes mail will be written to robotic tape and the disk space released.

In a subfolder called “archived documents” within the normal view of the user's active mailbox, the title, sender, date, and other information is still viewable. These documents are links to the archive database.

<sup>4</sup> Uwe Weist, Systems Engineer, Sun Microsystems GmbH. Uwe.wiest@sun.com.

Documents are automatically and transparently retrieved from tape if the user decides to open one of the items therein. Access time is less than a minute for archived documents using Sun 9840 drive technology and robotics such as the Sun L700.

Disaster recovery preparation and content duplication are handled transparently by the Sun SAM-QFS file system according to the customer's business policies. Lower storage costs, human resource efforts, and more reliable backups result in reduced TCO for customers.

### **UNC Chappel Hill<sup>5</sup>**

As one of the leading state universities in the nation, the University of North Carolina at Chapel Hill (UNC-CH) decided to implement an advanced group of major technology initiatives in the late 1990s. These projects, which included Internet distance learning programs, expanded in-class multimedia presentation equipment and an overall upgrade of the university's computing facilities, required major infrastructure improvements. Their existing mass-storage facility, a third generation UNIX system-based server running proprietary format data storage software, had serious bugs and was corrupting and occasionally deleting user data.

Under pressure to fix the problem, and with 3.5 terabytes (TB) of existing data, UNC-CH needed a solution that would allow users immediate access to their data in the proprietary format while the

systematic migration of data proceeded in the background. In addition, UNC-CH required complete transparency between the file system and the application as well as immediate and automatic data protection. Automated, flexible policy management, and scalable, high-capacity implementation were also important criteria in the software selection process.

To help with the project, the university contracted with Instrumental, a Minnesota-based high performance computing consulting group. After some preliminary tests, they demonstrated that it was possible to perform the conversion in-line by incorporating Sun SAM-QFS's Migration Toolkit software with a third party conversion application. Instrumental facilitated the in-line migration of UNC-CH's three million files to Sun SAM-QFS.

By working with a two-tiered, hard disk/tape storage HSM (hierarchical storage management) system, the data storage administrators are now able to optimize file organization. Sun SAM-QFS allows the system administrators to set data "high water marks" and archiving paths. It also routes the data automatically, following the pre-determined commands and ensuring that the data is appropriately stored for fast accessibility and/or long-term archiving. "Simplicity is one of the main strengths of our system," said Judson Knott, CIO at UNC-CH. "The software is based on a true file system that does not rely on databases to manage metadata."

UNC-CH's Odum Institute for Research in Social Science, for example, will take

---

<sup>5</sup> Judson Knott, CIO, University of North Carolina, Chappel Hill.

advantage of the new systems' higher capacity and file retrieval times to manage their vast social science data archives. The archive, which is the third largest of its kind in the U.S., is used by teachers and researchers in the field. The Digital Library is another virtual library project, which will include materials from UNC-CH, Duke, and North Carolina State University, as an on-line collection of databases, multimedia presentations, video and audio data, text, and digital images. The ever-expanding project requires huge data storage capacity.

Above these current projects, UNC-CH looks ahead to future initiatives. Beginning this year, all freshman students were required to have a laptop computer for online instructional material access, in-class multimedia presentations and other activities. The university's computing staff must facilitate this initiative by creating vast networks and intranets. They are also contemplating an expansive laptop storage backup project to allow the entire student body to upload the contents of their laptops onto a network for centralized storage.

UNC-CH's data storage and management system currently has over 175 TB of capacity and a state of the art hardware and software setup using Sun and StorageTek hardware.

### **Wright Patterson Air Force Base<sup>6</sup>**

Headquartered at Wright-Patterson Air Force Base, Ohio, the Aeronautical Systems Center (ASC) is the largest

research and development center of Air Force Materiel Command. Managing a wide variety of aircraft, missile, and related equipment programs, the ASC relies on a world class high-performance computing (HPC) environment that has the power to perform over 1000 gigaflops.

The ASC's HPC center user base includes almost 3,000 computational scientists and engineers at approximately 50 Department of Defense (DoD) laboratories and numerous university and industry sites. Projects such as the new F-22 Raptor fighter jet, the B-1 Lancer and B-2 Spirit bombers are managed at the ASC. The state of the art engineering required to develop these projects requires immense computing power.

To manage the immense quantities of data generated by researchers using the HPC system, ASC required a data storage system that could safely archive data for long-term preservation while also allowing quick access to files. In addition, ASC needed a system that was "fail-over safe," meaning that if one component of the data archive should fail, a separate component could revive the data and make it available to the network.

As part of the DoD HPC Modernization Program, the ASC decided to implement a SAN (storage area network) to manage its data archive. Comprised of two Sun Enterprise™ 10000 servers, three TB (terabytes) of hard disk cache and two StorageTek 9840 tape libraries, the system has the capacity to store 123 TB of data.

---

<sup>6</sup> Lisa Burns, Program Lead, Aeronautical Systems Center, Wright Patterson Airforce Base.

By creating a network of storage devices with fibre channel lines, the SAN's high-speed cabling and interfaces guarantee fast access to critical research data. Safe archiving was also assured, as the system was configured to store data redundantly across different pieces of storage media.

"SAM-QFS fit our system criteria by allowing fast access to the immense data archive and the ability to grow with our ever-increasing data quantities," said Vaughn Noga, director of integration and advanced technologies at the ASC. "The QFS file system has flexible archiving policies and can quickly stage files from the archive."

Last, the ASC administrators configured the data storage system to resist system failure. Because of the data's importance and the large number of researchers depending on the archive, ASC needed to ensure that the system was always on-line and available to the DoD community. To do this, they installed the data archive on multiple file systems distributed between two servers.

### **Australia National University<sup>7</sup>**

The Mass Data Storage System at the Australian National University (ANU) Supercomputer Facility provides a resource for data requiring storage and retrieval capabilities far beyond disk or workstation tape drive technology.

The Mass Data Storage system was acquired in 1993 to support supercomputer users and other types of data intensive projects. In mid-1997, ANU replaced their older storage

management software with SAM-FS from LSC for performance and reliability reasons. In the nine months that followed, ANU archived several terabytes of data. ANU's current storage configuration has a potential capacity of 300 terabytes.

### **BMW/Rolls-Royce GmbH<sup>8</sup>**

The BMW/Rolls-Royce GmbH division in Dahlewitz, Germany develops high-quality airplane-engines for private jets and small commercial planes. Several hundred engineers are employed by the company to refine the designs and create new innovations.

With a 50 terabyte data pool and with several hundred gigabytes of new data created each day, the company was running into major data storage problems. So much data was being managed that their existing backup hardware and software could not store the data fast enough. A traditional data backup could no longer be achieved in a reasonable time frame.

Working on a network, the engineers would save data to centralized tape libraries. However, a major data bottleneck was being created at the server. Bandwidth disparities between the network and the storage libraries were creating great inefficiencies in the system. While the tape libraries were capable of extremely fast data transfer rates, this bandwidth could not be fully utilized. Heavy network traffic combined with idle storage libraries pointed to the need for a more efficient

---

<sup>7</sup> [Http://anusf.anu.edu.au/MDSS](http://anusf.anu.edu.au/MDSS)

---

<sup>8</sup> Martina Pretzl, BMW Group, FI-34, Sapporobogen 6-8 Munchen

file management system and storage management software.

BMW began working with Synstar, a high-performance computing consulting group. Synstar installed Sun SAM-FS to work with BMW's existing Legato backup software. SAM-FS was configured to distribute the data at a bandwidth that would allow the tape libraries to function at near top speed. By intelligently segmenting the data for backup, SAM-FS collects user data from the network and distributes multiple data streams to the tape libraries. This allows many users to backup their data simultaneously, thus eliminating the data bottleneck. Sun SAM-FS allows the systems administrators to set data "high water marks" and archiving paths. It also routes and orders the data in a logical hierarchy, ensuring that the data is appropriately stored for fast accessibility and/or long-term archiving.

Overall operating costs have been lowered. With advanced data management software providing quick data access, BMW can decrease its dependency on large amounts of expensive online disk storage. With Sun SAM-FS in place, BMW is able to manage the vast quantities of data created each day. StorageTek and ATL tape libraries connected to two Sun Enterprise servers provide more than 50 terabytes of capacity.

### **Banta Corporation<sup>9</sup>**

In 1998, there was a growing demand from Banta's client base for a remote

digital workflow solution that would allow clients to manage their creative, prepress and print-processes. In response, Banta Digital Group, a division of Banta Corporation, began the implementation of B-real-time, a WAN (wide area network), accessible throughout the U.S. via high-speed telecommunications lines.

With hundreds of files and documents being updated or saved every hour, Banta's existing RAID and tape storage system would become incapacitated with the increased traffic levels brought on by the B-real-time network.

To streamline the data backup and security process, Banta implemented an efficient HSM (hierarchical storage management) configuration. By installing a Plasmon M500 magneto optical storage library, the clients' data could be proficiently and securely archived from online magnetic disk (RAID) to near-line optical disk (MO) and then finally backed-up onto off-line DLT tape.

Sun SAM-FS automatically and transparently copies files from expensive on-line disk to less expensive automated storage media, and restores the files back on-line as needed. SAM-FS automatically manages available disk capacities at specified thresholds by clearing disk space of copied data. Working with the enterprise server and the three levels of the storage hardware hierarchy, SAM-FS provides terabytes of secure storage capacity to B-real-time network users.

---

<sup>9</sup> Kenn Drapp, System Administrator, Banta Digital Group.

## **German Remote Sensing Data Center**

The Deutsches Fernerkundungsdatenzentrum, or DFD, is a division of the German Aerospace Center DLR, which is the German equivalent to the United States' National Aeronautics and Space Administration. The DFD is responsible for the acquisition, processing, archiving and dissemination of data from Earth Observation (EO) satellite missions of national interest or as a partner to international space agencies like NASA, ESA (European Space Agency) or ISRO (Indian Space Research Organization). The images and measurements made by optical, infrared, atmospheric and RADAR sensors onboard satellites are transmitted to a globally distributed network of ground stations, then processed, stored in long term archives and distributed to scientific evaluators and other users.

The raw data and the resulting products must be archived and catalogued so that they can be easily retrieved after minutes, days, weeks or even years. DFD plans to archive remote sensing raw data and products for long-term retrieval, this means over a period of at least 30 years. This is necessary to perform long-term analysis e.g. Long-term climate changes. The data may be retrieved via network or on a distribution medium (CD-ROM, DLT, Exabyte, etc.). The user information services can be accessed via a client server system (ISIS) and WWW interfaces.

Requirements for the DFD archive were for approximately two terabytes in 1996,

20 TB by 1999, 200TB by 2005, and ultimate capacity for at least a petabyte of storage. SAM-QFS was selected for this project in 1996 after struggles with another vendor's storage software.

## **Experience Music Project<sup>10</sup>**

About EMP - Experience Music Project (EMP) was conceived by Microsoft co-founder Paul G. Allen and his sister Jody Patton. The Seattle-based museum is a one-of-a-kind music museum combining interactive and interpretive exhibits to tell the story of American popular music. With their grand opening in late June 2000, EMP features a world-class collection of more than 80,000 music artifacts, state-of-the-art technology, and exciting interactive presentations. (For additional information on EMP, visit <http://www.emplive.com/>)

With over 80,000 digital artifacts, EMP needed a data archiving system to handle their multimedia assets. In addition to an extensive recorded sound archive, film, photographs, fanzines, and rare song sheets, EMP's artifacts include huge video clips as well as triple redundancy of all files.

The data archive was created for preserving EMP's vast collection of historical music artifacts. Also, a half Terabyte data cache would be available for keeping a large amount of data readily accessible for the developers working with the ever-changing audio/video media in the museum. By automatically and transparently copying files from EMP's disk cache

---

<sup>10</sup> Roger Jensen, UNIX Sysadmin, Experience Music Project, Seattle WA

(1/2 Terabyte StorageTek) to their automated DLT tape storage libraries, the software gives the appearance of an infinite disk.

## **About The Author**

*Lance Evans (lance.evans@sun.com) is a Systems Engineer in the SAM-QFS file system group within Sun's Network Storage division. He has tested and implemented high performance storage hardware and file systems for twelve years. He currently provides storage subsystem configuration, performance tuning, capacity planning, and architectural consultation to Sun's field force and to Sun's customers using the Sun SAM-QFS file system. Mr. Evans received a BS in Telecommunications from Ohio University in 1985, and prior to joining Sun has been employed by LSC, Storage Technology, Lockheed Engineering and Sciences, and Network Solutions.*

(C) 2002 Sun Microsystems, Inc. All rights reserved.

Sun, Sun Microsystems, the Sun Logo, Solaris and Sun Enterprise are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

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