

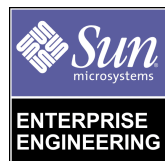


# Storage Area Networks A blueprint for Early Deployment

---

*Brian Wong - Network Storage Group*

*Sun BluePrints™ OnLine - January 2001*



<http://www.sun.com/blueprints>

**Sun Microsystems, Inc.**  
901 San Antonio Road  
Palo Alto, CA 94303 USA  
650 960-1300 fax 650 969-9131

Part No.: 816-0082-10  
Revision 01, 03/22/01  
Edition: January 2001

Copyright 2001 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303 U.S.A. All rights reserved.

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

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Sun BluePrints, SPARCstorage, Jiro, UltraSPARC, Starfire, Sun StorEdge, Sun Enterprise, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

**RESTRICTED RIGHTS:** Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a).

**DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.**

Copyright 2001 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, Californie 94303 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, Sun BluePrints, SPARCstorage, Jiro, UltraSPARC, Starfire, Sun StorEdge, Sun Enterprise, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REpondre A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



Please  
Recycle



Adobe PostScript

# Storage Area Networks

## A blueprint for Early Deployment

---

### Abstract

Storage Area Networks (SANs) are the subject of intense investigation in data centers around the world. Unfortunately, SAN technology has not advanced at the same rate as the associated marketing, and there is a substantial gap between user expectations and the ability of real users to deploy SAN technology. This paper surveys the applications to which SANs aspire, the available SAN technology-and its limitations-and attempts to prepare users for forthcoming technology, so that they can deploy real operational storage in data centers without further delay.

---

### Reasons for SANs

Storage Area Networks are intended to address a wide variety of issues in the deployment of relatively large quantities of storage. Sun plans to address these applications in a phased approach, beginning with the least risky (and least complex) and gradually attempting more and more sophistication. Viewed collectively, it's easy to see why users want to deploy SANs: they have the potential to solve many problems. However, the software—particularly *management* software—is not yet available to effectively deploy comprehensive solutions to many of these problems, especially when combined.

In the near- to medium term, SAN applications can be divided into the following categories:

- Management of distant configurations, adding flexibility to long-wave GBIC (LWGBIC) functionality.

"Soft recabling," the ability to logically redeploy storage from one host to another without requiring the movement of physical cables. In this mode, each unit of storage is accessible to at most one processing system. (In this context, a processing system may mean a cluster of cooperating hosts.)

"Storage consolidation," meaning the ability to make use of a large pool of storage amongst a wide variety of (possibly heterogeneous) hosts. Storage consolidation extends the soft recabling functionality by permitting multiple processing systems to use a single storage unit. This is *much* more complex.

Heterogeneous connectivity, usually meaning the ability to connect multiple brands of processing systems and multiple brands of storage. Although not the simplest configuration, basic data transfer connectivity between units is possible; more sophisticated interoperability, such as compatible management of various multipathing solutions, is *far* more complex.

Data sharing, meaning the ability of multiple hosts to access a single set of shared data, despite possibly dissimilar in basic instruction set, byte sizing or ordering, or operating system.

Massive configurations, specifically the ability to connect essentially limitless amounts of storage to a single processing system. Although the limits are reasonably high today (95 TB on a Starfire™ (Sun Enterprise™ 10000) using 18 GB disks), the value here is primarily in changing the ratio of consumed PCI slots to connected storage.

LAN-less and/or server-less backup, or the ability to perform backups without directly transferring data over a public IP network, and ideally with no impact on the processing systems. These ideals are far from being accomplished, although interesting gains in efficiency *can* be achieved even now.

These are the primary applications envisioned by Sun for storage area networks in the near future. This paper describes the technology used to implement these functions. In addition, we will call attention to some of the limitations of the technology as it exists today. Finally, we make recommendations as to how to choose among options in each technology category.

Although SANs appear to be a minor evolution in storage configurations, this greatly underestimates the technical situation. In limited configurations, SANs are quite similar to traditional direct-connect storage. However, the *extent* of a network configuration can have a qualitative impact on the design and implementation of network components. Storage devices have historically been designed assuming that they will be attached to a single host computer. The fact that storage devices have been successfully deployed in clusters is, for the most part, an artifact that "one-ness" is not significantly different than "two-ness." Certainly, two-node configurations are quite different from fully configured networks with many possible host-to-array connections. Large, complex configurations involving hundreds or thousands of entities involve many different considerations.

Within the devices and operating systems, many implementation choices are driven by the fundamental assumption of low-order connection counts. For example, command queues are sized to handle the load anticipated from a single (large) host; configurations can easily overwhelm typical command queue depth, leading to complex error recovery scenarios. Security is "handled" by assuming that the attached host operates in a secure way. In a multiply connected SAN, this security model is likely to be insufficient, because multiple hosts by definition cannot be trusted to cooperate smoothly. Tape drives are inherently single-system devices, although some SAN functions are specifically intended to permit tape transports to be shared among multiple hosts.

Fortunately, there is no reason that design decisions driven by the single-system assumption cannot be altered in future or sometimes even current products. However, understanding these limitations is crucial to avoiding them in a successful SAN deployment.

Because most existing devices and hosts have one or more single-connection limitations, the choice and management of SAN topology is undoubtedly the most significant implementation choice that the SAN user can make. Much of the discussion that follows identifies limitations of nodes in a SAN and focuses on topologies that permit the user to work around these limitations yet still accomplish the SAN's application goals.

---

## Topologies: Loop vs. Fabric

One of the most significant technology choices in FibreChannel SANs is the addressing mechanism. There are two primary addressing schemes, known as *private loop* and *fabric* mode. The technical difference is primarily the size of the address field associated with each node on the wire. Private loop uses a sparse 8-bit address field, providing a maximum station count of 126 nodes. (The address field is sparse due to the use of specific bit pattern combinations that permit faster recognition by switches and other components.) Fabric mode uses a 24-bit address, providing 16 million discrete addresses.

To some degree, the addressing scheme isn't interesting to end users except in how it affects the potential extent of the SAN. In private loop mode, the complexity of the loop configuration is not materially different from that of simple hub configurations. In particular, it is feasible to exhaustively probe the loop to determine what devices are present (even if this is a slow process). In a fabric with more than 16 million ( $2^{24}$ ) targets, this process is no longer even remotely feasible. (Each device may also have many LUNs—potentially hundreds or even thousands—making an exhaustive probe even longer!) Rather than requiring that each interested party attempt to exhaustively probe the fabric, the switches maintain a *name service*, structurally

similar to the name services (such as DNS) provided for IP networks. Interested nodes-in practice, host bus adapters and switches-query the name service to determine the identities and addresses of connected devices.

Probably the most important single characteristic of private loop mode is that every loop is treated as a network that has no connection to any other network. In the event that a user configures topologies more complex than a simple loop, unusual things happen. In the case of having two parallel private loops connected to a single system (for example for path redundancy), the device drivers will not recognize the fact that the disks are duplicate images and two logical devices are created in / devices for each physical drive. Switch zoning (or, ideally, multipath device drivers) must be employed to eliminate or reconcile these confusing and potentially problematic multiple images.

---

## Fabric Implications

The presence of the name service means that the HBA must be instructed to query the name service in order to report the identity of the attached devices. The name service is responsible for reporting the current state of the fabric when queried, as well as notifying interested parties when nodes enter and leave the fabric. Nodes can change state for a variety of reasons. The most obvious example is a plug event, where a cable is plugged into or removed from a switch port. However, changes in fabric zoning also change the connectivity between a device and an HBA, as do changes in the state of remote links.

One of the requirements of moving to fabric mode is that every attached device must be capable of handling both the 24-bit address and be name-service aware. This is difficult enough for relatively large and capable fab-ric entities such as switches, HBAs (which typically require new device drivers) and storage processors, but this can be excruciating for small edge devices such as disk drives and tape drives. Note that using fab-ric mode requires new device drivers for the HBAs that are aware of and can make use of the fabric name service. The StorEdge Network Foundation driver stack delivers fab-ric-aware drivers. These drivers will only be supported on 64-bit platforms (Solaris 7 and higher), so configuring a host on a fabric mode SAN requires upgrading to Solaris 7 or Solaris 8. The Foundation drivers stack can coexist in a single system with earlier drivers; to distinguish between the two, the HBA must have special FCODE downloaded into it that causes the HBA to be identified as a fabric-capable device. (Existing FCODE causes the HBA to be bound to the existing,

non-fabric driver stack.)Typically, the switch to fabric mode requires new firmware or microcode. The logistics of such conversions are daunting. They are similar to the impending conversion from IPv4 to IPv6, with the complicating factor that a FibreChannel network can operate in either private loop mode or fabric mode-but not both.

In the future, a third type of topology will likely become more prevalent. Called *translated mode* by Ancor, this topology requires that the switch translate a fabric address to a private loop address. In order to identify the difference between otherwise identical 7-bit private loop addresses, each private loop is assigned a prefix address, permitting unambiguous addressing in the fabric. This means that the devices can operate in private loop mode, yet still participate in a full fabric interconnection scheme. Unfortunately, this functionality is not standardized across switch vendors, although more or less complete convergence appears likely over the next year or so. In the meantime, consider carefully whether or not to deploy fabric in each segment-avoid complexity.

Disk drives configured directly on a fabric require fabric-aware firmware. Although this is technologically possible, it is a complex undertaking. The number of different versions of firmware and their widely varying provenance means that having fabrics using many directly-connected disk drives are likely to be a logistical headache. In addition to the fact that each directly addressable drive consumes a fabric address, managing the address spaces and microcode versions is likely to be problematic.

This problem leads to one of the simplifying decisions in deploying a SAN. Sun does not expect to deploy JBODs on fabric-mode SANs. Preparing for fabric mode deployments will require the upgrade of JBOD storage to controller-based array technology. An example of this might be to use a Sun StorEdge™ Data Management Center as a fabric-aware aggregator for existing JBODs such as A5x00 JBODs. An alternative to interposing controllers between JBODs and fabric mode SANS is to separate these units onto what amount to separate SANs operating in private loop mode. This eliminates most of the operational issues.

This strategy is likely to be required for other devices, such as older disk arrays that are not fabric-aware. In the near term, this presents a challenge for management software, because it effectively divides a single physical SAN into different logical "islands." However, this is a temporary problem, since the problem is equivalent to managing a global storage pool across a set of disjoint zones in a full fabric.

---

## Other Software Considerations

Fabric implementations also impose secondary requirements on host software. Probably the most significant of these is the dynamic nature of a large fabric. Networks of devices will change configurations at a fairly rapid pace: new devices

are added to the configuration; old ones are removed. More importantly, the visibility of various devices may change, even if the physical configuration remains constant.

An example of this might be a change in zoning: moving a zone boundary can make a given device visible to a host that previously did not see it, while also removing visibility from another, previously-used host. In these cases the host operating systems must be able to react to the changes. Most operating systems (including Solaris OE) are only able to handle these device state changes with manual intervention-some others require a reboot. On Solaris, reconfiguring the device state is accomplished by issuing either a `drvconfig; disks; devlinks` command or using `luxadm(1)` to re-probe the device address space. A framework should be wrapped around the device driver infrastructure so that changes in the fabric name server can be automatically reflected in the operating system's device address space. Such a framework will need to avoid the use of loop initializations (LIPs), in order to minimize the disruptions caused to other devices by the LIP process.

---

## JBODs vs. Intelligent Arrays

Related to the problem of upgrading disks to fabric capable firmware is the issue of controlling the sheer complexity of SAN configurations. This is greatly exacerbated by the "simple" nature of JBODs. JBODs expose the intricacies of every disk drive to the entire fabric. The switched nature of a fabric mitigates this to some degree, because the switch divides the logical network into multiple independent loops. When a switch connects a host and a disk as in the diagram below, the host-to-switch and switch-to-disk are on completely separate FCAL loops. At the FibreChannel level, the disk and host do not communicate directly. Any problems caused by the peculiarities of either unit's interface to the FibreChannel are isolated. Signal integrity or loop integrity problems fall into this category. Diagnosability is good, and corrective action is generally easy.

Unfortunately, the operational realities are not so simple. There are actually two completely separate conversations taking place. Although the switch partitions the FibreChannel conversation, there is *also* a host-to-peripheral conversation being conducted at the SCSI level, and of course this crosses the switch's loop boundaries.

This is crucial, because the host's device driver must be able to handle the peculiarities of the peripheral's command and especially status replies. Although nominally mandated by the SCSI command set standard, responses-and especially timing-vary subtly between drives and between versions of firmware. This is particularly true of target drivers such as `sd`, `ssd` and `st`.

JBODs place stringent logistical demands on a fabric, because they expose every version of microcode in every type of disk drive, potentially to all participating hosts. Because the device drivers must carry implicit or explicit timing and other dependencies on the specifics of the remote devices, it is important to limit the exposure of driver-to-firmware interactions. Historically, each model of disk drive goes through ten or fifteen different versions of firmware during its product life, although any individual site will probably only see a quarter of this number.

Combined with the sheer number of models of disk drive, a large SAN would likely have a staggering variety of firmware versions. Recall that the same type of disk drive such as "10,000 RPM 9 GB" is normally sourced from two or three vendors, and that each of these sources has a different firmware base. Moreover, vendors often provide different versions of the "same" disk drive in order to reduce cost, leading to even more varieties. Disk drives are generally installed and useful for several years, so a typical data center will accumulate dozens or even hundreds of different versions of firmware. This will be exacerbated by the introduction of complex fabric-aware protocols into the firmware code bases.

Because of these issues, Sun recommends against configuring JBODs directly into fabric configurations. To be clear, there isn't anything that will prevent the use of JBODs in fabrics-and a casual test will obviously show that "it works." However, over time such configurations will become difficult to manage. Instead of direct attachment, Sun advises that JBODs be connected to *fabrics* via a disk array controller. In such configurations, the array controller operates the host connection in fabric mode. The back end loops, which connect the array controller to the disk drives, are operated in the much simpler private loop mode.

This is not to say that JBODs are not useful, only that they shouldn't be used in fabric mode. Using JBODs in private loop configurations does not create the level of exposure, since there are only 126 possible target addresses, most of which are consumed by the disk drives themselves. Practically speaking, this limits the number of hosts that can interact with each set of disk firmware.

The use of private loop configurations also limits the amount of storage that can be connected to individual hosts. Private loop configurations have limited addressing, so complex topologies involving routing and multiple switches aren't interesting. In all probability, truly massive configurations will require the use of fabric implementations.

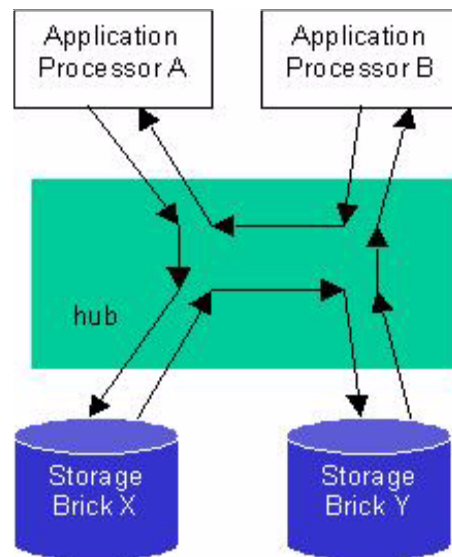
---

## Switches vs. Hubs

As with Ethernet, FibreChannel network offerings include both hubs and switches. In IP networks, the primary difference between hubs and switches is the availability of bandwidth. With hubs, all connected entities share the bandwidth of a single

segment. In contrast, switches provide each branch with dedicated node-to-switch bandwidth. The same is true for FC switches, but this is *not* the primary short-term benefit of deploying FC switches.

When compared to simple hubs, FibreChannel switches provide a substantially improved technology for error isolation within a connected domain. More specifically, hubs connect all nodes on a *single* logical loop; an error can easily be isolated to the loop, but more precise location of errors is difficult. Switches place every port on a different logical loop, making it much easier to identify the location of a signal integrity fault.



**FIGURE 1** FibreChannel hub topology

Heterogeneous configurations (in this case, meaning multiple sources for processing systems and/or storage systems) should gravitate toward switched configurations rather than shared loops configured with hubs. This is due to the complexity of interacting storage microcode and operating systems: since switches provide a simple mechanism to isolate unrelated configuration subcomponents from each other. Given an HP server connected to Sun storage, a Sun™ server connected to HDS storage and an NT server connected to EMC storage, all sharing a switch, it's possible to isolate each of these conversations from each other, thus preventing (for example) reset and error conditions from affecting unrelated units.

Unfortunately, while switches are more useful than hubs, switches are far more complex than hubs-and much less reliable. This is due to the fact that hubs are simply electrical repeaters: they copy a signal from one wire to another one. A few hubs also retime the signal, but even this function is done exclusively in hardware. In contrast, a switch not only copies frames from a loop to the switch backplane and

to another loop, it must also decode the *contents* of the frame, for example to implement zoning, routing, and similar functions. All of these functions are implemented with firmware, leading to a far more complex unit.

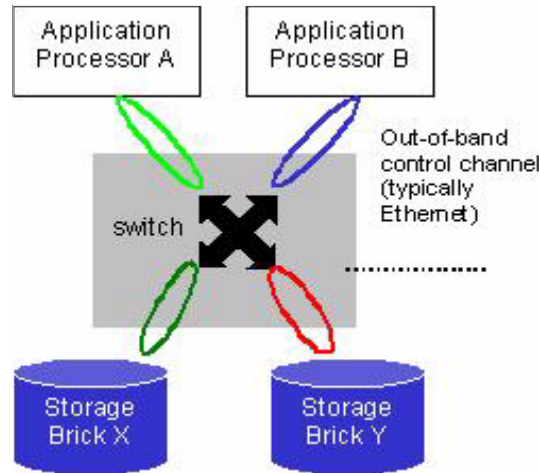


FIGURE 2 FibreChannel switch topology

Most users seem to be planning to use switches exclusively in SAN deployments, possibly because of similar practices in the IP networking arena. Sun recommends that users carefully consider the use of both hubs *and* switches in their deployments, due to the substantially lower complexity and higher reliability of hubs. In general, use switches when the fabric must be logically divided, while hubs should be used for increasing connectivity. The primary reasons for dividing the fabric are for zoning and creating error domains for debugging purposes.

Error domains are important in large configurations because they are the "width" of the configuration that is subject to a single error. In FibreChannel, the error domain is a single physical loop. Although this is important in IP (or any other) network, the relative immaturity of FibreChannel makes management of the size of error domains particularly important. One of the most troublesome problems in FibreChannel implementation is the *loop initialization protocol* (LIP). Similar to a full bus reset in parallel (electrical) SCSI implementations, LIPs cause substantial "shock" to every device on the loop. Devices treat LIPs differently, but in most cases, LIPs cause command queues to be reset, timeouts to be reinitialized, and similar drastic actions. It is therefore particularly useful to separate logically related storage into separate loops or error domains. For example, VxVM mirrors should be configured onto different loops, and global hot spare should be on yet different loops, thus isolating and minimizing the effects of LIPs. In some cases, configurations with many initiators have been observed to create "LIP storms" in which performance drops to nearly zero during error recovery on multiple hosts and HBAs.

It is important not to over-emphasize the RAS implications of using switches; although some of these functions are available now, many others are not yet implemented. Seemingly simple functions such as identification of actual topology can be difficult or even impossible. For example, identifying a loopback configuration is difficult at best, despite the fact that this is easy to create with a single wrong connection.

---

## Distance

One of the key uses for switched configurations is in the area of extended distance. This usually arises in disaster recovery or business continuance applications, where one (or more) leg of a mirror is physically placed at campus- or even short-haul metropolitan distances of a few kilometers from the processing systems. Although it is feasible to directly connect such mirrors using hubs and long-wave GBICs (LWGBICs), this stretches the limits of the FibreChannel technology. In fact, Sun has in the past found it necessary to mildly restrict LWGBIC configurations for precisely these reasons. For example, the standard maximum configuration for local connections between systems and the Sun StorEdge A5x00 Healthcheck permits four A5100 or three A5200 enclosures per loop. Using LWGBICs, only three A5100s are permitted per loop, due to the lack of signal integrity margin at full distances when fully configured.

The technical constraint in this case is the physically large size of the loop, exacerbated by a simple architecture. In these configurations, the signal loss can occur anywhere in the loop; since there is only one loop, it can be difficult to identify the problematic cable or connection. Switches offer a greatly superior solution. A long-distance configuration involves placing the long-haul connection between two switches; this arrangement divides the overall topology into at least three discrete loops. Signal integrity problems can be quickly isolated between the "fenceposts" created by the switches. Moreover, the number of arrays attached through the single HBA is no longer limited by timing and signal integrity margins on the loop. Arrays can be configured without specific constraints on the remote switch, because they are on independent loops.

This arrangement also has the advantage of making it easier for vendors to qualify and support a wide array of configurations. In the future, long-wave connections will be supported *only* via switches, permitting all arrays and all hosts to be immediately qualified for long-haul operation without having to test a full matrix of interactions.

Sun recommends that users plan to deploy switches at each end of campus-distance configurations, in order to simplify cabling and improve diagnosability. As with any other mission-critical application, laying a pair of fibre cables for each switch-to-switch configuration is strongly recommended. Furthermore, laying *four* fibres per

pair of switches is also recommended, due to the disproportionately high cost of labor in the deployment process. Although they will not be used in initial distance deployments, subsequent phases of rollout will utilize multiple switches with redundant paths to further guarantee data availability.

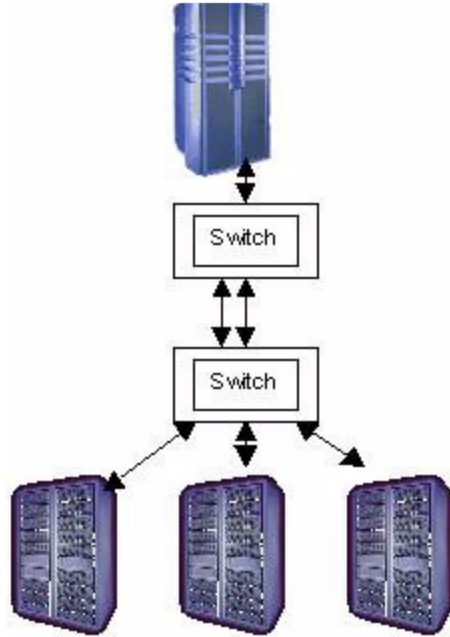


FIGURE 3 Using switches to gain distance

---

## Bandwidth

FibreChannel switches also provide some opportunities to improve performance. Because they divide a logical network into physically discrete segments, it is possible to use switches to improve the bandwidth between storage and hosts. In bandwidth-intensive applications such as decision support, data mining, image (or movie!) manipulation and seismic processing, switches can sometimes improve I/O performance by eliminating loop bottlenecks. Of course, this requires multiple HBA connections and some attention to the bandwidth of the centerplane of the switch. As with Ethernet switches, most FibreChannel switches have enough bandwidth to provide many simultaneous full-bandwidth connections-but not enough to provide full bandwidth every pair of ports.

It is worth noting that few-if any-file server, database transaction processing or general-purpose applications consume very much I/O bandwidth, largely due to a combination of low I/O content (per unit of processing work) and the small size of the I/Os that is inherent to this class of applications. For example, a fully configured Starfire Server running at 100% utilization moves less than 250 MB/sec across a 40+ TB storage farm while running the TPC-C benchmark, despite an unusually high I/O content. This represents about 4 MB/sec per I/O channel. Similarly, a 24-processor E6500 moves about 100 MB/sec of I/O during a high-end LADDIS (NFS) run. This I/O load is spread across 12-15 disk arrays, or about 5-8 MB/sec per I/O channel. For comparison, an 8-way E4500 running a decision support application can easily consume more than 300 MB/sec, spread across fewer arrays. Even if the access is spread across only six arrays, bandwidth per link is only 50 MB/sec, necessitating bandwidth improvements only in the event that access is highly skewed across the arrays. (Thin-wide striping, or plaiding, would manage this problem away.)

Sun recommends measuring or estimating the data movement involved in an application (even one of the data intensive applications sited above) before opting for a fully switched configuration, due to the relatively low data volumes that are *usually* involved.

---

## Storage Consolidation

Storage consolidation is a term that covers a wide range of applications. In general, it refers to the ability to have a single pool of storage that is shared by a relatively large number of processing systems. There is increasing incentive to do this because storage density is increasing quite rapidly.

Most of the requirements come from smaller systems, generally from larger workgroup-sized systems. These 2- to 8-processor systems often have aggregate storage requirements that are much smaller than typical disk array capacities. As a result, users have a strong motivation to find ways to connect several systems to a single array, thus eliminating the "fragmentation" effect created by dedicating a minimum-capacity disk array to each system. ISPs (and xSPs in general) are often the most sensitive to this problem, since they tend to have the largest number of small systems. Large systems (16-way and up) usually require enough storage that they entirely consume multiple disk arrays, in spite of high capacity disk drives.

Fragmentation is also an issue when allocating for peak consumption. Most storage administrators allocate a substantial amount of excess storage to each application—perhaps as much as 30%—against the possibility that the application will suddenly consume more disk space than normal. This might be due to seasonal or otherwise unpredictable changes in the surrounding business environment, or for technical reasons such as temporary file space allocated on a non-linear basis.

Storage consolidation reduces fragmentation of peak consumption space by permitting redeployment of storage quickly and without recabling. This is most practical when redeploying within an array, although some topologies permit more flexible arrangements. Because storage space allocation is a manual operation that is not supported by enforcement software, the reduction in total cost of ownership due to minimizing fragmentation is lower than corresponding elimination of excess processing capacity in server consolidation. (Host resource managers—including Solaris Resource Manager™—enforce administrative allocation of CPU processing in servers. At present, there are no corresponding local or global mechanisms for allocating access to storage resources, in or out a network.) Nonetheless, reduction of peak storage space consumption is a meaningful goal even without supporting administrative software.

Generally speaking, users feel that storage consolidation is an easy thing to do, based on casual experimentation. Given switches (or even hubs), it is quite feasible to connect multiple hosts to a disk array and transfer data effectively with all of them. However, this isn't a complete test, and it does not reveal limitations in current software. One must be cognizant of these limitations in order to properly deploy storage consolidation efforts in the near future.

The biggest issue has to do with controlling I/O rates that arrive in a single disk array. Each host maintains an I/O queue to each target in its device driver; targets correspond to disk array controllers. When the connection is dedicated—meaning that only one active system at a time can operate on the array's data—the host is able to control how commands are queued to the array. In particular, the array and the host have the same view of the I/O load.

When the I/O stream into the array comes from multiple sources, the array does *not* have the same view of the stream as (any of) the hosts. This leads to difficulties in managing queue, in particular handling queue-full conditions. This is especially problematic for certain edge cases, because not all SCSI commands<sup>1</sup> are treated equally; some of them, for example, REPORT\_LUN, are much less tolerant of timing delays than others (in fact, they are quite *intolerant*). There isn't anything intrinsically impossible with managing these queues, but there are many details that must all be handled correctly, especially error conditions.

For currently deployed device drivers, the only solution is to strictly limit the number of I/O commands that the drivers will transmit to the array. Each array has a maximum pending I/O queue depth. For Sun arrays (A3500FC and T300), this limit is 255 pending commands; some arrays are higher, others, such as the current EMC Symmetrix, appear to be lower (64 pending commands per channel director). The objective is to arrange matters so that there is no realistic possibility of filling the queue. For an array with depth 255, the reasonable maximum is 250 commands. These 250 pending commands can be divided as appropriate between sharing systems.

1. FibreChannel implementations use FC as a transport mechanism for SCSI commands and responses.

For example, if four systems are attached, one might permit each of the four systems to queue as up to  $250 / 4 = 62$  commands. This can be implemented by adding "set `ssd_max_throttle=62`" in `/etc/system`. There is no need to allocate evenly. An important system might be permitted 235 commands, while the other three might be permitted only 4 each. Be aware that there is no mechanism to prevent each system from accidentally over-configuring `ssd_max_throttle` to, say, 220. With four systems, this would theoretically permit 880 outstanding I/Os against the controller queue. In the event of heavy load on two or more systems, the queue on the array may fill, resulting in errors. Two unrelated systems performing a database checkpoint might well cause create such a circumstance.

Setting the pending command depth may seem to be a serious limitation, but in fact few arrays are pressed to the maximum queue depth, so this is rarely an issue with single systems. The queue depth for smaller systems such as Ultra-60s and E220Rs generally will not exceed 5 pending commands, even at full CPU utilization. These systems can be consolidated onto a single array without anticipation of much trouble. However, measure systems before attempting to consolidate them! A few applications generate very unpredictable I/O load, and consolidating these may result in circumstances that cause very deep queues and attendant problems in the array controller. These problems affect all hosts accessing data through the overloaded controller.

A more permanent solution to the queuing problem is the use of adaptive throttling in each of the host device drivers. This feature is under development for `ssd2` driver, but it will not be available until late 2000.

In the event that a controller is overloaded with work (not necessarily the same thing as having a full queue), storage systems encounter another limitation: arrays do not have any resource control mechanisms that permit an administrator to prioritize I/O access in the event that the controller or its resources (internal busses, caches, HBAs or disk drives) become overloaded. Consolidating multiple systems' I/O load onto a single array also consolidates the risk of missing performance goals across all attached systems, because arrays generally do not make it easy to identify the source of the I/O requests. Although controller capability is the most obvious shared constraint, it is much more likely that interference between systems is contention for cache resources or command queue availability, rather than sheer CPU (or XOR engine) resources.

For these reasons, the most suitable near-term candidates for storage consolidation are those with relatively low I/O rates. Fortunately, this generally corresponds to smaller systems, because I/O rates are strongly correlated to system processor performance. (This is intuitive but not obvious: I/O is required to feed data to the processor and to sink data generated by the processor. Accordingly, systems with slower and/or fewer processors generate much less I/O than systems with many and/or faster processors.) A system is a good candidate for storage consolidation now if its global I/O rate is less than about 200-300 logical I/Os per second<sup>3</sup>, if the

2. The `ssd` driver is used by Sun's FibreChannel disk storage, notably the A5x00 family, the A3500FC, and the T3. Other storage typically uses the `sd` driver.

array is to be shared by two hosts. If the array is to be shared by four hosts, the systems should average 100-150 logical I/Os per second. These I/O rates are generally consistent with typical loads observed on dual- and quad-processor systems such as Sun Enterprise™ E220R through Sun Enterprise E450 Servers.

Some applications are notable for their high I/O content and are consequently not good candidates for consolidation at this time. In particular, email hubs tend to have substantially higher than average I/O content, on both average and instantaneous basis. This is unfortunate, since email also tends to be relatively undemanding of storage capacity. The other notable application with higher-than-typical I/O content in the small-system space is a software development hub-especially a build server that is shared amongst many developers.

On the other hand, most low-end database servers and most file servers in general tend to have low-to-average I/O content, and thus make good candidates for consolidation. Web servers, especially those serving typical *intranet* populations, tend to have the lowest I/O content of all, and make excellent candidates unless they *also* host databases. As with server consolidation, training and test systems are generally the best initial candidates for storage consolidation. Other less important functions should be consolidated before primary, mission-critical systems are considered.

---

## Consolidation of Boot Devices

Boot disks must be treated differently, especially in the area of storage consolidation. Although it is both theoretically possible and operationally appealing to consolidate boot disks into a few restricted areas of a SAN, this is unlikely to be technically feasible in the near to medium term. This arises in the context of large configurations such as Starfire systems, where as many as sixteen domains each has discrete requirements for boot facilities.

From a purely logistical perspective, a simple SAN island operated in private-loop mode can be used as a boot device. This is effectively the same as a hub- or directly-connected disk, such as an A5200. There are no technical issues associated with these simple configurations. Bootability of devices is not affected by configuration on private loops; if the device is bootable when directly connected, it is still bootable on a private loop. However, few storage consolidation configurations use these simple configurations. Note that zoning is likely *not* useful in this scenario, because the individual drives in a JBOD enclosure usually cannot be divided into different zones.

3. In this context, "logical I/O" refers to I/Os against a RAID volume, rather than I/Os to the member disks. This can be measured by observing `iostat(1)` for hardware RAID volumes and Solstice DiskSuite™ metadisks, or by `vxstat(1)` for VxVM volumes.

The most effective consolidations will likely occur using full fabric mode, combined with some form of LUN masking/management. These environments are *much* more complex, and are unlikely to permit boot capability. The primary reason for this is that the boot process occurs without the full cooperation of the operating system. On Sun systems, the Open Boot PROM (OBP) is used to control the system during the boot and crash dump processes; in particular, it provides a limited-capability device driver for SCSI and FibreChannel host bus adapters. If a boot device were to be configured on a fabric, the OBP's HBA driver would have to be modified to login to the fabric, query the name service, use fabric addressing and handle the widely varying latencies associated with multi-hop switched configurations. This isn't a simple process, especially given the very restricted facilities available to the OBP drivers.

Moreover, a consolidated fabric environment is far more complex than a simple, directly attached boot disk. In the event of a failure, debugging a failing boot process is likely to be difficult without some of the services normally provided by the operating system. Sun advises that boot devices be kept as simple as possible. Do not expect to be able to consolidate boot disks in fabric implementations for quite some time, due to the complexities of implementing OBP-level failure analysis/recovery and fabric login.

---

## Soft Recabling

Sometimes storage consolidation is taken to mean that a set of storage is to be redeployed on a cyclic basis, without necessarily presuming that the storage will be actively serving more than one of the candidate systems at any given moment. This function is better called "soft recabling," than "storage consolidation" and is far less problematic-and risky-than many-to-one storage consolidation efforts. Soft recabling amounts to the ability to switch the host-to-storage connections logically, rather than physically. Of course, this requires the use of a switch, and almost certainly the zoning functionality in a switch.

As long as the switch is capable of zoning, soft recabling is quite easy. However, deploying soft recabling on a large scale (say, a data center the size of Sun's Enterprise Technology Centers-190 hosts and 220+ TB of storage in 160 racks) requires some careful planning of topology. The issues are not complex, but do require attention to detail. Soft recabling requires that switches be deployed as "dividers" between functionally independent groups, so it pays to deploy switches between anticipated "storage cells." This can be complicated by the general recommendation that hubs be used as the expansion mechanism; entire hub-connected loops are therefore obliged to be in a common pool that is allocated and recabled as a group.

Soft recabling is problematic without zoning in the switch. It is certainly possible to adopt usage conventions that ensure that the storage is only accessed by the "allocated" hosts. However, this overlooks some key details in the implementations. For example, each HBA that is visible to the array will consume two SCSI *initiators* within the array's tables. The size of the tables varies by array and firmware version, and some arrays run out of table entries at fairly low levels: most cannot handle 32 initiators, and many cannot support any more than four or eight. Sun recommends avoiding soft recabling topologies without zoning switches.

---

## Server-less and LAN-less Backup

One of the more interesting uses for soft recabling does not involve disk storage at all. This is the so-called "server-less" or more accurately, "LAN-less" backup. Backup is an onerous task, and one that can consume an enormous amount of resources on a regular basis. The overhead of backup motivates users to find a way to avoid or preferably eliminate this extra work.

In an ideal world, the backup is simply a function of the storage fabric, occurring without the knowledge of the processing systems. Additionally, transferring the data from disk to tape media avoids crossing public IP data networks.

Only some of this can be achieved using software available today, due to the processor-storage architecture currently in use. Specifically, the unit of exchange between processing systems and storage is a *device*; the only semantics ascribed to the device are that bytes are logically numbered from 0 to  $n-1$ . In particular, no application level semantics are inherent in the description of the device. This means that it is impossible to determine whether or not a given byte belongs to a data file, a database table, a file's metadata, or something else (perhaps a file system's log) without using knowledge that is specific to the processing system that manages the data in the first place.

As a result, most backups will *still* involve the processing system to a significant degree. When backing up a file system, only the processing system understands the on-disk format of the file system, so it is the only entity that is able to identify where files begin and end, and which files are interesting to backup at any given moment. Backup software runs on the application processing systems to read the file system or database and identify which blocks must be backed up. The list of blocks is then transferred to a data movement engine external to the application processor. The data mover is then responsible for obtaining the actual data blocks from the (nominally shared) storage and writing them to a suitable tape media.

It is important to understand that the process of identifying the block list is a substantial fraction of the work associated with actually performing the backup. This is especially true for incremental backups (the most common type), because

they tend to move relatively little data. In the worst case-an incremental backup of a file system with files that average a single block-the "serverless" backup is about the same amount of effort as a traditional backup! The larger the files to be backed up, the more overhead that is saved by moving the backup into the SAN.

Even when a SAN data mover performs the backup, the backup must be done with the explicit cooperation of the processing systems, because the backup must take care to make a consistent copy of the data. The server-less backup strategy will usually be combined with point-in-time copy functions such as Instant Image, to minimize the cost of ensuring consistency.

Another consideration when planning server-less backup is the utilization imposed on the source disk drives. Because disk arrays have no means of expressing priority of I/O, even a server-less and LAN-less backup has the ability to severely impact the performance of a processing system by creating very high utilization and consequent queuing for disk resources. For this reason, independent copies are strongly preferred over dependent copies, if this is operationally feasible. Note that this is particularly true of incremental backups, because they tend to impose a much higher ratio of seeks to data transfers than epochal dumps.

Once the block list is generated, the data mover is responsible for transferring data from disk to tape. Because the data mover uses a FibreChannel SAN to obtain the data and transfer it to a tape, the data is not transmitted over a public IP network, and is therefore regarded as "hidden." The primary values of "LAN-less" backup are the displacement of the data over a private network and especially the elimination of TCP/IP overhead for source-to-sink transfers. Note that public networks are often just as fast as the back-end FibreChannel SAN networks: gigabit Ethernet has essentially the same media speed as currently deployed FibreChannel at 1 Gbit/sec. The work associated with transferring 100+ MB/sec over a TCP/IP network is considerable, consuming multiple UltraSPARC™ processors on both client and server systems. Transferring the same data over a SAN avoids the TCP/IP overhead, freeing these resources for application computing.

However, even LAN-less backup isn't without issues. Backups, especially epochal backups, can transfer massive volumes of data at very high speed. When many large and fast tapes (such as STK Redwood, STK 9840 or IBM Magstar) are used, most modern backup software is capable of moving 2 TB-3 TB/hr (approaching 1 GB/sec), through a single large data mover such as an Sun Enterprise E6500 Server. (At these rates, the data mover's centerplane becomes the limiting factor.) This is the most important application in which the additional bandwidth available from switched topologies is meaningful. Plan backup topologies carefully to avoid bandwidth contention.

One significant operational advantage of FibreChannel over SCSI is the connection of tape drives. With sustained performance of most high-end tape transports approaching 20 MB/sec, common parallel SCSI implementations-typically 20 MB/sec for burst transfers-are becoming problematic. Faster versions of SCSI are

available, but they trade transfer speed for configuration distance. Ultra 80 SCSI implementations require low-voltage differential (LVD) connections, and at a maximum distance of just 12.5 meters.

FibreChannel tape interface is far more practical, because FC has far more lenient distance restrictions (kilometers instead of meters), combined with much higher raw bit rates. Unfortunately, FC tape transports are still immature, and require special configuration attention. Operational difficulties like handling LIPs are common. Many tape transports are unaccustomed to being configured on shared interconnects, and often will reset or even rewind in the event of an unexpected LIP. If the LIP is unrelated to the tape transport, a rewind operation would be extremely disruptive. As a result, FibreChannel tape transports should be configured on isolated loops, shielded from untoward interference. Given the magnitude of a disaster in the event that backups are destroyed or improperly written, the safest configuration for tapes is to use point-to-point topologies, and not risk connecting the FC tapes to the general SAN at all. If FC tapes are connected on a general-purpose SAN, administrators must be absolutely certain to properly maintain zone configurations.

A related topic is the consolidation of tape transports, using techniques and motivations similar to that of basic storage (disk) consolidation. Many sites wish to use fewer, faster tape transports, and then to share them among several backup hosts or SAN data movers. Unfortunately, given the inflexibility of FC tapes, Sun cannot recommend this practice today. Even using zoned switches is probably not sufficient in the general case, since a mistake in zone configuration can have substantial consequences on backup operations. Eventually, tape and library management software will be integrated with zone management, making this a much safer organization.

---

## Management Software

One of the most pressing issues throughout the SAN environment is an industry-wide lack of sophisticated management software. Each of the various SAN functions has a GUI (the switches have GUIs, arrays have GUIs, the limited LUN management software has a GUI, etc.), but integration of the various functions is seriously lacking. This causes Sun to recommend that, for the near term, SANs be deployed only with limited scope. In the event that large-scale SANs must be implemented, the most realistic approach is to divide them into smaller "islands" of management. Managing topology, for example, is difficult to impossible, both because each switch has differing management mechanisms and because topology is a higher-level function than either the switch or the wires that make up the topology. This is one of the technical issues behind the Jiro™ Technology initiative—it enables the creation of compatible management functions, potentially across functions that are widely varying in semantics.

One of clearest examples of the management software problem is the management of consolidated storage. Ideally, storage consolidation provides the administrator with a single global pool of storage resources that can be configured to any of the hosts in the administrative domain. However, today there is no software that permits management of storage residing in multiple arrays; as a result, each array must be managed as a separate pool, largely defeating the administrative benefits of sharing storage in the SAN. (Obviously this does *not* reduce the benefits of eliminating fragmentation and sheer reduction in acquisition costs.)

Furthermore, in the long run, it will be important to manage interconnect topology in conjunction with the storage allocation policies, especially for RAS considerations. However, topology and allocation are presently unrelated functions and do not integrate at all. Several vendors are working on this functionality, but it will be some time before useful integration will be achieved.

---

## Management Interoperability

A major consideration when deploying FibreChannel SANs is the fundamental incompatibility between the management infrastructures of the various switch vendors. Most (all?) switch vendors are trying very hard to add unique value into their products. Since the FibreChannel specification is fixed, their best opportunities for adding value are in management.

This provides useful opportunities to put new functionality into the marketplace, but it also means that virtually every management function on every switch is different and incompatible with others. Zoning control on a Brocade switch is different from zoning control on an Ancor switch, for example. Eventually, Jiro Technology management facades will be constructed around most majority SAN entities, and these will close up many of the detail differences between individual units. Unfortunately, this process will take substantial time, and in the meantime, users must be prepared to separately manage many different types of SAN hardware and software.

---

## Data Sharing

Perhaps the least mature SAN application is heterogeneous data sharing, in which more than one processing system is granted access to a single (possibly replicated) storage image. For the most part, this application is not even implemented, due to the complexity of translating data formats between dissimilar processing systems. Even homogeneous data sharing is far less sophisticated than one might expect.

In principle, this seems to be an easy application to manage. Just hook up the storage to two (or more) processing systems, mount all but one of them read-only, and data can be shared. Unfortunately, it's not so easy.

Virtually all file systems use caching to improve performance. Even file systems such as UFS and VxFS that have direct I/O options apply these only to user data; they all cache metadata such as indirect blocks and directory entries. Once metadata is cached on a system, it can become obsolete (and therefore inconsistent) through modification on the writeable copy. This problem is also true of the journaling systems used by most file systems to avoid long reboot times, since data can be stored in either the journal (log) or the master, and failure to obtain the correct set will result in apparently corrupted data. In order to effectively share a file system between multiple processing systems, the common file system must either be completely read-only or protected by a data sharing protocol similar to the lock managers used by NFS and CIFS.

A few vendors are beginning to offer single-writer, multiple-reader file systems that can be shared on interconnected storage. In addition to depending on resolution of all of the various issues associated with storage consolidation (effectively this function is similar to consolidating two systems into one, from a storage perspective), the robustness, performance and scalability of the consistency protocols remains to be proven. Virtually all of the protocol issues are the same as those addressed by existing network file systems such as NFS, AFS and CIFS.

There is no magic in the fact that the storage is shared, only that access to the data is likely to be more efficient than over traditional IP-based networks. The efficiency is not gained by having high-speed storage interconnects; instead, it is gained by avoiding the CPU-intensive TCP/IP protocol processing.

Fortunately, proven solutions already exist for most data sharing requirements, in the form of NFS. These solutions have already addressed most of the consistency, security and locking issues that must be handled in the directly shared storage environment. Contrary to common belief, I/O rates in many-if not most-shared data applications are not especially high, and in particular are well within the capabilities of even small-scale NFS servers.

One very common application for shared storage is the distribution of master content to a large number of front-end web servers. If the site sustains 100,000,000 hits per day, the maximum page retrieval rate is 1150/sec, even assuming that every hit requires a retrieval from the central NFS server. We know that the data transfer operations make up 37% of the load in the SPECsfs97v2 benchmark; this means that the 1150 hits/sec roughly corresponds to a SPECsfs97v2 score of  $1150 / 0.37 = 3108$  NFSops/sec. Since a 2x400 MHz system<sup>4</sup> scores 5952 ops/sec with a latency of 3.96 ms, a uniprocessor system is almost capable of handling this load. This analysis is extremely pessimistic, because the clients will certainly cache a large fraction of the incoming hits, in either their main memory caches or using a CacheFS

4. The cited result is from a 2x 400 MHz/4MB E3500. NFS is not sensitive to CPU cache size, so a comparably equipped E220R or U60 would likely outperform the Sun Enterprise 3500 by a small margin, due to decreased memory latency.

on-disk structure. This would reduce the actual data rate to within the abilities of a system similar to an Ultra 10. Note that 1150 hits/sec represents an aggregate rate of less than 15 MB/sec, even assuming relatively large 13 KB hits as used by the SPECweb96 benchmark (high-end sites normally take strict measures to reduce hit size and consequent Internet bandwidth).

In another example, a tier-1 national ISP recently investigated shared data SAN solutions for providing consistent directory data to a number of systems supporting several hundred thousand users. The directory lookup rate was approximately 2,000 per second; each lookup transferred about 100 bytes, corresponding to a data rate of approximately 200 KB/sec—well within the capabilities of even a 10baseT network. With a NFS demand rate of  $2000/0.37 = 5405$  ops/sec, this application is also well within the capabilities of a low-end NFS server. In this case, there is very little benefit obtained from client-side caching. An objective analysis of required data rates is required before committing to a solution, but in general, most data sharing activities appear to be at a controlled level that can be addressed by file service protocols such as NFS or even CIFS.

Of course, the use of NFS is not a panacea, since large sites such as these will require that the data services be continuously available via a clustered solution. Sharing a mirrored RAID volume is administratively much easier than an NFS cluster. On the other hand, NFS is widely deployable on an immediate basis across heterogeneous processing systems. For these reasons, Sun currently recommends the use of NFS for most data sharing requirements while data sharing techniques mature.

---

## SAN vs. NAS

Data sharing and incremental backup are representative of a set of applications that require that storage functions be in possession of certain application-level semantics about the data. This is the primary architectural difference between Storage Area Networks and "Network Attached Storage (NAS)."

Although there is no inherent reason why this must be the case, SANs are customarily implemented using a device protocol; the file system or database is owned by the processing system. The only information the storage system has about the data is its device host and offsets. In contrast, NAS systems are universally implemented using a *file* level protocol: the storage owns the file system, and the storage (file) server maintains metadata such as file size, access attributes, last access times and exported name. In addition, file servers typically hide the fact that they typically do *volume management*. Whether or not the NFS server uses simple or compound RAID volumes to store its file systems is not interesting to a client. The only exposed concepts are about the file itself, rather than about the underlying storage.

The unifying concept between these two approaches is that a NAS system uses device services to provide storage. That is today, every NAS system uses either directly attached storage (or in a few sites) a SAN. In fact, a NAS system can be viewed simply as a protocol converter: the public protocol is the file protocol (NFS or CIFS); the internal protocol is the SCSI command protocol as applied to block devices. There is no reason that a single suitably equipped SAN could not offer both file- and device- level services.

---

## Security Issues

Another thing to consider carefully in SAN environments is security. When storage is directly connected to individual processing systems, interaction between the storage and any foreign processing system is arbitrated by the owning system(s). Usually it takes a fair amount of effort to get through the security systems of most commercial operating systems. However, when all storage is interconnected by what amounts to a single dedicated network, security becomes a much more important issue.

In this case, each disk array is responsible for its own security. In the past, disk arrays have not been responsible for security, and most arrays have no provisions for asserting that a given host or host bus adapter may or may not have access to the array, LUN or data set. Although some arrays have "LUN masking," enforcement is spotty and management is not common to all such arrays.

Switches are responsible for some types of partitioning between storage domains, notably by zoning incompatible uses from each other. Although zoning is hardly an industrial-strength security measure, it is the primary defensive mechanism available in SANs today. Maintaining the security of the switches in a SAN is crucial to protecting the SAN's data. This is especially true given the relatively primitive security provisions found in most switches. In most, administrative security is protected only by a super-user password. Unfortunately, these passwords are usually transmitted over a control network in the clear (i.e., not in encrypted form), meaning that the control networks must be kept under careful management.

---

## LUN Masking

LUN masking is a technique for specifying how LUNs and/or devices are to be made available to specific hosts. Most LUN masking implementations are offered by array vendors, but there exist multiple viable ways to implement this. The disk array approach requires that the administrator specify to the array which hosts (actually

which world wide names) have access to each LUN, and the array enforces the specification. This strategy is useful in applications that include Windows NT systems. By permitting only limited access right from the array, NT systems are prevented from unexpectedly taking control of (and possibly formatting!) all visible LUNs in the fabric.

A different form of the same function uses a layered device driver (usually called a "shim") placed in each host. A centralized directory provides information to the shims; each shim permits its host to access to only LUNs that are so prescribed in the directory. This implementation is able to work with any type of array (including arrays that do not have LUN masking), but has the drawback that it requires a shim on every host member in the SAN. Failure to ensure the presence of the shim could lead to conflicting access if there is no compensating arrangement made in switch zoning. The most common shim implementation of LUN masking is from HP/Transoft (available on NT and Solaris OE, shortly on HP-UX).

Finally, some switch vendors have proposed putting the LUN masking in the microcode of various switches. None of these implementations are available yet, but they promise to combine the generality of the host-based shim with the security of the array offerings. Of course, every version of LUN masking administration is completely different-and incompatible.

---

## Separate, Dedicated Control Network

Most users think of SANs as being based on FibreChannel, and this is certainly true today. However, there are actually two distinct information streams in a SAN. The obvious stream is the data flow, with data being carried by FibreChannel between storage arrays and processing systems. Modern arrays (and processing systems) are increasingly relying on separate control networks dedicated to management functions. In the past, storage management was handled in-band, meaning that command and control information was transmitted over the same logical and physical infrastructure as the data itself. The vast majority of deployed disk arrays (A3500FC, SPARCstorage™ Array, etc.) use in-band management.

New arrays, such as the Sun StorEdge T3 family, rely on out-of-band management. These arrays typically carry management commands over a separate interconnect, usually IP-based. Although FibreChannel networks are perfectly capable of carrying IP traffic, there are sound technical reasons why out-of-band mechanisms are becoming more widespread. The most compelling reason is the ability to use a separate path for diagnosis purposes. In the event that an array stops responding to data requests, a diagnosing entity can attempt to use an alternate path to the array to determine whether the data path has failed, or if the entire array has become

disabled. Ethernet networks are sufficiently unlike FibreChannel networks that it is quite unlikely that independent failures will occur simultaneously on the data and control paths.

Out-of-band management is also provided on FC switches, data service platforms, and even some managed hubs. Servers are increasingly equipped with diagnostic and/or console processors—for example the service processors found in Starfire servers—and these also require dedicated control networks.

Data centers are strongly counseled to provide control networks that are physically separated from networks that handle data, especially public networks. This is particularly important given the current relatively low level of security available from many service processors (and even servers). In the likely event that remote management must be provided, active measures such as firewalls and IP packet screens should be taken to ensure that only authorized traffic is able to use the control network.

---

## Final Comments on Topology

In the final analysis, SAN technologies are essentially being used to separate logical traffic topologies from the physical interconnection topology. Because of the myriad technical issues, zoning is *essential* to success once a SAN grows past the size of existing loop configurations. In fact, most large configurations have essentially configured a myriad of logical point-to-point connections; these logical connections are carried over a physically continuous interconnect. In essence, this is soft recabling carried to a very large extent.

One major installation uses in excess of 400 discrete zones to manage the storage demands of about sixty Solaris images. Interestingly, there are only slightly more storage devices (around 500) than zones!

Because of the critical nature of the topologies, it is well worth the effort to apply rigorous change control procedures to the zone configurations. Errors in zone management can have enormous impact on the viability of the entire interconnected configuration. In fact, as a result of early adopter experience with SAN technology, most large sites that are installing SANs are in fact configuring *two* parallel SANs. In these configurations, the hosts and storage are each connected to independent SAN fabrics. Multipath drivers balance I/O across the two fabrics. This arrangement permits an operation to survive a catastrophic event in one of the fabrics.

---

## Summary

After several years of development, SAN technology is finally stabilizing at a level of functionality that permits users to derive limited benefit from new, more varied configurations. However, the complexity of SANs, combined with the lack of maturity in some areas of SAN software means that early adopters must tread carefully. To date, most SAN development efforts have been confined to specific individual functionality. This isn't a bug in development terms—one has to walk before running—but it does mean that it will be difficult to deploy full-strength, multi-application SANs in anything like the reasonably smooth, enterprise-quality that people wish.

Accordingly, the approach that most users should take is one of caution. Decide clearly which SAN functions are of use in each specific environment. Most applications appear to benefit primarily from one, rather than many, of the primary SAN functions. In the event that your data center can benefit from more than one function, carefully consider whether or not these functions should be configured together on a single (logical) SAN; if so, consider how much effort will be required to manage multiple overlapping functions. In particular, avoid SAN deployments that involve data sharing.

Topologies must also be carefully considered. The first order of business is to identify useful boundaries for error domains and zoning domains. These boundaries will generally dictate the choice between hubs and switches and the actual topology to be installed on the floor. At the same time, this permits optimization of capital outlay—switches cost *much* more than hubs.

In general, complex topologies are presently both difficult to manage and difficult to deploy, since many storage products are not fabric-compatible. Moreover, complex topologies—especially those involving multiply cascaded switch configurations—are presently risky, because they rely on zoning to constrain the exposure of devices to various problematic situations such as unexpected LIPs and initiator counts. Fortunately, the restricted size and scope of current SAN functions do not require full fabric implementations.

Acquisition for SANs obviously involves procuring interconnect infrastructure such as hubs, switches and cabling. At the same time, it is wise to begin deployment of a secure parallel management/control network, for both storage and processing systems. In the event that a large-scale fabric implementation is appropriate, substantial capital must be allocated for upgrading components that are not fabric-aware, such as older arrays controllers, JBODs and bridges.

Consider carefully when storage consolidation will be appropriate, and whether or not storage consolidation really means extending the operating domains of individual arrays or simply soft recabling. Soft recabling can be deployed

immediately, given suitable FibreChannel topologies. True storage consolidation is unrealistic before the arrival of modified device drivers that use more sophisticated queuing and flow control algorithms due later this year. True consolidation should be limited by internal initiator constraints to at most four to six hosts at any given time. Plan the data center topology so that no more than four to six hosts are located in any given topological cell.

Perhaps the most salient significant task for potential users of storage area networks is to carefully plan the topology of a data center network fabric. Especially if immediate deployment is necessary, connected topologies may be difficult to organize due to the relatively limited addressing of private loop configurations. Most of the various SAN applications require fairly peculiar—and often different or conflicting—topologies.

Most importantly, understand the limitations of the various technologies, as significant differences exist in almost every case between the industry hype and the present deliverable products. Plan to work around these shortcomings while the industry further develops the technology.

---

#### *Author's Bio: Brian Wong*

*Brian Wong is a Sun Distinguished Systems Engineer specializing in systems architecture, including capacity planning and deployment strategies for high-end server and storage configurations.*

*Brian is the systems architect for the Network Storage group. Prior to joining Network Storage, he was chief scientist for Sun's Computer Systems Enterprise Engineering group.*

*Brian has held various positions in Sun since joining in 1987, including sales, marketing and product development.*