An Introduction to SAN Capacity Planning

Mark B. Friedman
Datacore Software
1020 Eighth Avenue South, Suite 6
Naples, FL USA 34102
markf@datacore.com

Abstract.

Emerging technology that allows the construction of high performance storage area networks (SANs) requires extending existing analytic frameworks that can accurately predict the performance of complex disk subsystems. This paper focuses specifically on two elements of emerging SAN technology: (1) the performance of Fibre Channel links and connecting hubs and switches, and (2) the performance of in-band and out-of-band SAN data management protocols. Traces and timings from benchmarking tests conducted with an in-band SAN data manager product are analyzed and discussed. Understanding this measurement data should enable existing disk modeling approaches to be extended to encompass this new storage technology.

Introduction

Storage Area Networks (SANs) are constructed around high bandwidth, low latency Fibre Channel (FC) connections between storage peripherals and computer hosts. Exploiting Fibre Channel technology, SANs can be built that interconnect large numbers of host computers and peripheral storage devices. In SANs, storage resources no longer need to be *tethered* to a specific host computer in an inflexible way that restricts access to those resources. Instead, in a SAN multiple host computers can gain direct access to a common, shared pool of storage resources.

Some form of SAN management is naturally necessary to mediate access to shared storage resources. This paper examines two crucial elements of emerging SAN technology: (1) the performance of Fibre Channel links and connecting hubs and switches; and (2) the performance of in-band and out-of-band SAN data management protocols. Ultimately, pulling all these elements into a rigorous conceptual framework for disk performance modeling, such as the one formulated in [1], should allow us to configure, tune, and manage the growth of high performance storage networks.

Because storage area networking using Fibre Channel technology is relatively new, it inevitably raises challenges to storage administrators responsible for their configuration and deployment. Fortunately, many of the elements of SAN capacity planning are familiar. Magnetic disk performance is a constant whether configured in a SAN or more conventionally attached to a host computer. Cached RAID disk subsystems, for another example, are also a core component of SANs, raising familiar issues of capacity vs. performance trade-offs. However, Storage Area Networking does introduce some additional hardware wrinkles, including optical Fibre Channel links and switched networking topologies that add a new dimension to the analysis. Understanding the performance of these new hardware components is a crucial first step.

Another new component associated with SAN technology that has a performance dimension is the SAN data management protocol. It has become commonplace to describe competing SAN data management protocols as being either *in-band* or *out-of-band*.

In this paper we will discuss traces and timings from an in-band SAN data manager software product called SANsymphony that executes on a conventional Intel server running the Microsoft Windows 2000 operating system. In-band SAN data management means that some SAN component actively participates in each disk data access operation, redirecting it either to Intel server memory (configured as a cache) or to the appropriate physical device. For a variety of reasons, in-band SAN data management is a superior approach to first generation, out-of-band techniques. Because its data management services are transparent to SAN clients, in-band SAN data managers are much easier to deploy than out-of-band approaches that normally require specific client file system or device driver software and/or hardware modifications. Moreover, as this document will discuss in detail below, relatively inexpensive in-band SAN data managers can provide superior performance using familiar memory-resident caching techniques. [2]

What is a SAN?

Storage Area Networks are created around Fibre Channel connections linking host computers and storage peripherals such as magnetic disk drives, tape, etc. Figure 1 provides a conceptual view of a SAN, according to this definition. Basically, some number of host computers are linked to a set of storage resources. How the host comput-

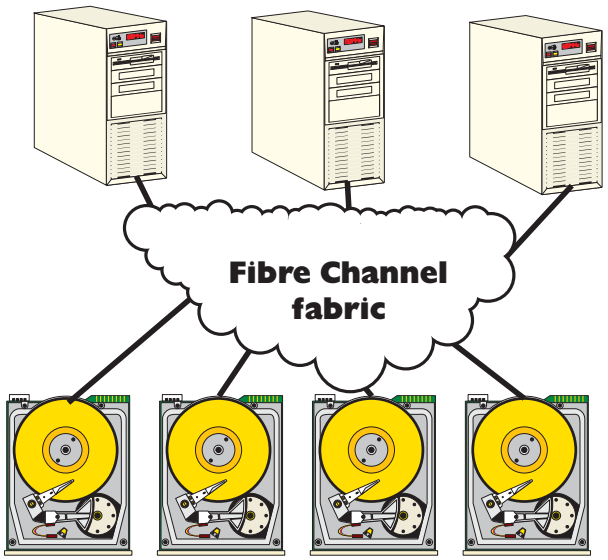


FIGURE 1. A CONCEPTUAL VIEW OF A SAN. IN A SAN, SOME NUMBER OF HOST COMPUTERS ARE LINKED TO A SET OF STORAGE RESOURCES.

ers are configured or how the storage resources they are attached to are configured is immaterial. What matters is that there is an interconnection fabric that links these components together, potentially allowing any host to address any storage device that is attached to the storage network.

It is not generally understood that installing Fibre Channel technology alone does not create a SAN. Fibre Channel merely provides the necessary plumbing that make SANs possible. Once multiple SAN clients gain access to a pool of shared disk resources, some form of data management protocol is necessary to mediate access between the clients and the storage devices. Some form of user identification and authentication is required to ensure that SAN clients only access the disks and/or files that they are authorized to access. If volumes or files are shared, a serialization and locking mechanism is also required to preserve data integrity. If physical storage resources are pooled and some form of virtualization is used to create the logical volumes and files that SAN clients access, a resource manager responsible for allocating and freeing physical resources is necessary. Virtualization also requires ensuring that each client's view of the mapping between logical to physical devices is current. These essential SAN data management functions all transcend the traditional facilities of Fibre Channel and its associated networking hardware.

SANs vs. NAS. SANs based on current technology have one additional key requirement, namely that the host computer clients on a SAN address SAN storage resources, such as disk drives, directly using the SCSI command set. This clearly distinguishes SAN technology from Network-attached storage (NAS) devices, with which they are apt to be confused. "SAN" may be "NAS" spelled backwards, but that coincidence is not something that helps to clarify the fundamental distinction between the two approaches.

In a NAS environment, storage devices are directly attached or tethered to specific host computers. These hosts then permit access to their directly-attached disks using network-oriented protocols such as nfs, CIFS, or http. In NAS, disk access requests are transformed using networking volume and file access protocols that also specify matters such as security access and file locking.

Conceptually, a NAS environment looks like Figure 2, showing two computers using networking protocols to establish remote disk volume and file access. A networking client establishes a connection with a file server using standard protocols such as SMB (aka CIFS), nfs, or http. These application level protocols run on top of the TCP host-to-host networking protocol that is responsible for setting up and managing the network connection between host computers for the duration of a session. Routing packets from one node on the network to another is handled by the lower level IP layer, which, in turn runs, on top of a Link Layer that interfaces directly to networking hardware.

Because they leverage an existing TCP/IP internetworking infrastructure, NAS devices are ordinarily easy to deploy. But remote file access mediated by traditional TCP/IP-based networking protocols has inherent limitations, which restrict the environments where NAS machines can provide acceptable levels of service.

Performance considerations distinguish NAS from SAN, favoring an interconnection technology specifically designed for storage. Never optimized for disk access, many aspects of the standard TCP/IP network protocol are far from ideal for storage devices. For example, the Ethernet protocol that furnishes the backbone technology for Local Area Networking (LANs) uses a maximum transmission unit (MTU) of 1500 bytes. Meanwhile, SCSI commands transfer data in block-oriented chunks of 4 KB or larger. Data intensive applications like digital video streaming often attempt to move data between the

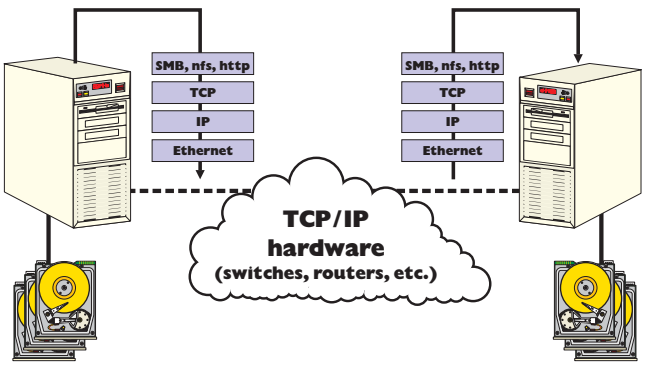


FIGURE 2. A CONCEPTUAL VIEW OF NAS. STORAGE DEVICES ARE DIRECTLY ATTACHED TO SPECIFIC HOST COMPUTERS. THESE HOSTS THEN PERMIT ACCESS TO THEIR DIRECTLY-ATTACHED DISKS USING NETWORK-ORIENTED PROTOCOLS SUCH AS NFS, CIFS, OR HTTP.

computer and disk in chunks as large as 1 MB. Breaking large chunks of data into MTU-sized packets (a function of the IP layer) for transmission and then reassembling small packets into larger application-oriented chunks is inefficient and time-consuming. The performance of NAS devices when the SCSI block size is not a good match to the underlying network MTU is notoriously poor. (For one typical example, see [3], especially section 9, pages 45-48.)

Another example is the acknowledgement mechanism that TCP Host sessions use to confirm successful delivery of each packet sent is unnecessary when computer hosts access disk devices. Disk channels are reliable connections that, compared to more transient networking connections, are quite unlikely to fail. Consequently, a significantly more efficient mechanism can be used that notifies the host only when errors occur. This explains why most of the current industry interest in direct network-attached disks revolves around utilizing the IP packet routing layer, bypassing the upper level TCP host-to-host layer.

Considerations such as these dictate that a new, storage-oriented infrastructure is needed to connect assorted computer hosts to assorted disks directly. Unlike NAS, SANs require constructing a new, separate and distinct networking infrastructure built around Fibre Channel links, adaptors, hubs, and switches. The SAN infrastructure conceptually duplicates hardware that currently exists to make LAN and WAN connections, an overlap in function that is only natural since the interconnectivity goals of both technologies are quite similar. (In fact, the designers of the Fibre Channel specification deliberately borrowed quite freely from computer networking technology.) This duplication of function is unavoidable, however, if SANs capable of virtualization with pooling of shared storage resources are to be constructed. Without belaboring this point any further, let's proceed directly to a discussion of the technical capabilities of Fibre Channel technology that make this new world of storage area networking possible.

Because both NAS and SAN technology have their place, a hybridization process has begun where elements of NAS and SAN are commingled, blurring the finer technological differences. Inevitably, we can expect hybrid devices that incorporate elements of both SANs and NAS. A NAS device that internally or externally accesses Fibre Channel disks is one form of hybrid. Or, what shall we call a networking client lacking direct SAN connectivity that can still access a SAN client's virtual disks using networking protocols and hardware links. In another hybridization example, a high availability SAN storage manager cluster relies on TCP/IP connectivity to communicate status information to other nodes in the cluster.

Fibre Channel technology

Fibre Channel (FC) is a relatively new, standards-based interface technology that functionally replaces SCSI to link host computers to disk, tape, and other peripherals. Compared to the older SCSI parallel bus hardware technology that it was designed to supercede, the Fibre Channel specification provides the following extended features:

- Support for high speed serial connections over optical fiber and copper wire connections
- Support for extended distances (up to 120 km on multi-mode fiber)
- Extended device addressing (up to 27 or 128 addresses on a Fiber Channel-Arbitrated Loop segment; up to 224 or 16,000,000 unique addresses on a multi-segment FC fabric)
- Redundant channel connections for reliability and failover
- Dynamic, hot-pluggable optical interfaces

These extended capabilities of Fibre Channel hardware sparked interest in developing storage area networking (SAN), with the goal of seamlessly allowing host computers to access pooled storage devices sharing an interconnection fabric. Naturally, once there is direct connectivity between multiple hosts and storage devices, it is no longer possible to rely on host-centric data access and control protocols. This leads to the requirement to develop native SAN data management protocols that perform command and control functions consistent with a distributed environment. We intend to explore the performance implications of SAN data management protocols in some detail in the body of this paper.

SANs also introduce new hardware that provides the connectivity functions to link multiple host computers to pooled storage resources. Managing the many physical connections between computers and the storage pool is the province of switching and routing hardware analogous to LAN hubs, switches, and bridges. Understanding the performance characteristics of this new link technology and how that performance is impacted by hardware connectivity options is a crucial new aspect of SAN configuration planning. Below, we will attempt to remove some of the mystery associated with FC by characterizing the performance of this new hardware.

Connectivity

The Fibre Channel specification supports three connection topologies: point-to-point, arbitrated loop, and fabric. Point-to-point connections are simple links between one FC controller and another. For the sake of wiring convenience, Fibre Channel hubs are available that simplify point-to-point interconnectivity. Fibre Channel-Arbitrated Loop (FC-AL) connections support a dual loop configuration (for redundancy and failover) capable of accessing up to 127 target addresses on a single loop. Switched networks, or the fabric topology, is by far the most popular topology, mainly because it supports extended addressing to up to 16,000,000 attached devices. The potential for attaching that many host computers and

storage devices to the network accentuates the need for extended data management services for these potentially very large environments.

Switches are the main class of FC network gear that is used to link hosts and devices to a fabric. Similar to Ethernet switches, FC fabric switches provide dedicated virtual circuits between ports with the full Fibre Channel bandwidth available concurrently for each virtual circuit in use. In one vendor's switch implementation, for example, shift registers associated with each port are connected to a high bandwidth crossbar circuit. After establishing a virtual circuit between input and output ports for the duration of a packet transmission, bits that register on the inbound port are promptly echoed on the outbound port. This kind of high speed FC switching hardware where the complete virtual circuit is confined to a single switch adds only about a microsecond of latency to an FC packet transmission.

Multiple switches can be interconnected or cascaded to extend the scope of the fabric beyond the number of individual ports that a single switch contains. Switching a packet transmission between cascaded switches introduces additional complexity due to (1) the potentially longer distances involved (each km of distance adds about 5 microseconds latency to the transmission) and (2) the potential for contention on Inter-Switch links (ISLs). Due to the myriad ways that switches can be interconnected, the performance analysis of cascaded fabric switches grows quite complex. Although a full discussion is beyond the scope of this paper, interested Readers can review an informative discussion of this subject in [4].

FC Protocol

Serial protocols look like bit streams on the wire. It is common practice to transmit bit streams in recoverable segment called packets, and Fibre Channel is no exception to being packet-oriented. Similar to the OSI 7 layer model that influenced the design of the TCP/IP Internet protocols, Fibre Channel supports a set of well defined layers that operate on packets, as illustrated in Figure 3, which is adapted from the official specification.

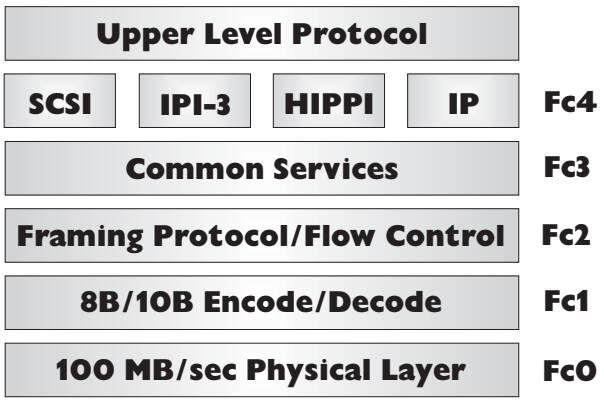


Figure 3. The layers of the Fibre Channel protocol.

It should be noted that FC protocol design decisions reflect fundamental differences between storage area networks and conventional TCP/IP-based networking. FC's deliberate divergence from cherished internetworking standards developed with painstaking care over the last twenty years is due to the need to operate networks for storage devices efficiently. Transmissions between host computers and peripheral storage devices demand both high throughput and low latency; they need to be optimized for those requirements. Moreover, the networking links used to connect host computers and peripherals, even in fiber-based storage networks that support dynamic and non-disruptive reconfiguration, reflect persistent, durable, and reliable connections. (Host computers generally do not tolerate transient state changes that would make a file system periodically appear and disappear.)

Serial protocols can accommodate much higher transmission rates than the parallel bus technology used in SCSI that FC supercedes. During the arbitration phase of SCSI bus command processing, for example, an electrical control signal must travel up and back on the parallel hardware bus and allow time for target devices to respond. The arbitration phase on a SCSI bus lasts approximately 500 microseconds before the SCSI command initiator and target are latched. Command processing is followed by a clean-up phase that quiesces the bus in anticipation of the next transaction. This involves similar bus signaling that also lasts about 500 microseconds. Delays of this duration were tolerable when the SCSI bus ran at 5, 10 or 20 MB/second. At higher transmission rates, however, more and more of the SCSI bus bandwidth is consumed by this handshaking delay. In addition, the lengthy electrical signaling needed in SCSI to establish and tear down connection was eliminated in the Fibre Channel serial protocol.

In contrast to TCP/IP, Fibre Channel pushes packet flow control deep into the interface hardware. The FC2, or Framing Layer, is responsible for generating packets of the required size for outbound traffic and for re-assembling them on the inbound side. Packets from multiple transmissions can be interleaved, providing for very high effective data transmission rates. The Framing Layer runs in the interface hardware (both in the Host bus adapter and at the target device), rather than utilizing an external software stack. An advantage of this approach is that it reduces the number of interrupts that attached hosts must process. Rather than generate a host interrupt each time a frame is received, a Fibre Channel adaptor need only present one interrupt to the host computer at the conclusion of a successful Read operation.[1]

[1] In contrast, high-speed networking (gigabit Ethernet or faster) is plagued by the overhead of interrupt processing. See, for example, [5]. Each Ethernet packet invokes processing by multiple layers of the complex TCP/IP software stack. Processing a 1 Gb Ethernet link using standard 1.5 Kb frames would theoretically generates 80,000 host computer interrupts per second that would require processing by the full TCP/IP software stack.

Another throughput-oriented efficiency results from the FC Framing Layer's guaranteed in-order delivery of packets. In-order delivery greatly simplifies the buffering logic for the node processing the inbound data stream. Guaranteed delivery in a switched fabric ensures that packets are never routinely dropped whenever there is contention. The reliable, in-order packet delivery mechanism of FC also eliminates the need for both (1) TCP-style acknowledgement packets that have an acute impact on effective data transmission bandwidth over extended distance links and (2) complex retransmission time-out (RTO) logic in the software stack that are necessary to deal with an unreliable delivery service like IP. [6]

As Figure 3 above indicates, the lower Fibre Channel levels were designed to support multiple upper level protocols, including both SCSI and IP network traffic. Theoretically, transmitting IP packets over FC links is one way to achieve interoperability of FC and IP networks. In addition, there are currently several industry initiatives attacking the interoperability problem from the opposite direction by encapsulating SCSI commands in IP packets. The ultimate goal of both exercises is to let a single optical wiring infrastructure carry traffic to and from both types of devices. The convenience factor of having hybrid network gear that can handle both IP and SCSI traffic appeals to organizations facing the cost of installing a new, optical fiber infrastructure. One sobering assessment of this inevitable convergence is that the performance of hybrid networks is apt to be less than optimal based on the need to process two workloads with starkly different characteristics. Applications that demand the highest levels of storage or networking performance in this brave new world will be better served when SCSI and IP traffic are confined to dedicated sub-networks.

Fibre Channel performance

The role Fibre Channel technology plays in SAN performance is relatively straightforward. The Fibre Channel protocol increases the performance of typical SCSI command processing by implementing faster links. It also eliminates the lengthy SCSI bus arbitration phase. The result is a peripheral connection technology that delivers both high bandwidth and low latency. This section explores these two aspects of Fibre Channel performance.

To ease the transition to a new interface technology, Fibre Channel fully supports the existing SCSI command set. In a SAN, SCSI commands are encapsulated in a packet-oriented protocol designed to support high speed serial links. Currently, the most common form of Fibre Channel technology utilizes optical fiber links running at 100 MB/sec. Recently, higher speed 200 MB/sec links have been introduced. It is also possible to run the Fibre Channel protocol over copper wire for short distances. The timing and test results discussed here were all obtained using 100 MB/sec optical FC connections.

Figure 4 below illustrates the timing for a typical 16 KB Read request from a SAN client over a 100 MB/sec Fibre Channel link to a SANsymphony Storage Domain Server (SDS). The timing data here was acquired using a Finisar hardware monitor. A complete 16 KB Read request is performed in approximately 330 µsecs. The command execution is broken into a number of different frames processed in three distinct phases, as illustrated.

Initially, a small frame containing the SCSI Read command is processed. The SANsymphony SDS software explicitly responds to the SCSI Read request, translating the virtual SCSI disk address known to the client into a physical address for an SDS-managed disk. The software then searches the SDS data cache for the data requested. On a cache hit, data transfer can begin immediately, and no further delay in processing the Read request is necessary. The Command processing phase lasts approximately 140 µsecs from the initiation of the request to the first data transfer frame that is returned by the SDS in reply. We will explore what happens during that 140 µsecs of processing inside the SDS in a moment.

The Command processing phase is followed by a reply that initiates the data transfer phase. This consists of 16 1024 byte data frames that are processed approximately every 10.2 µsecs, which corresponds to the 100 MB/sec instantaneous transfer rate that first generation FC links support. The block data transfer phase is followed by a frame that contains the SCSI status that marks the command complete. There is a

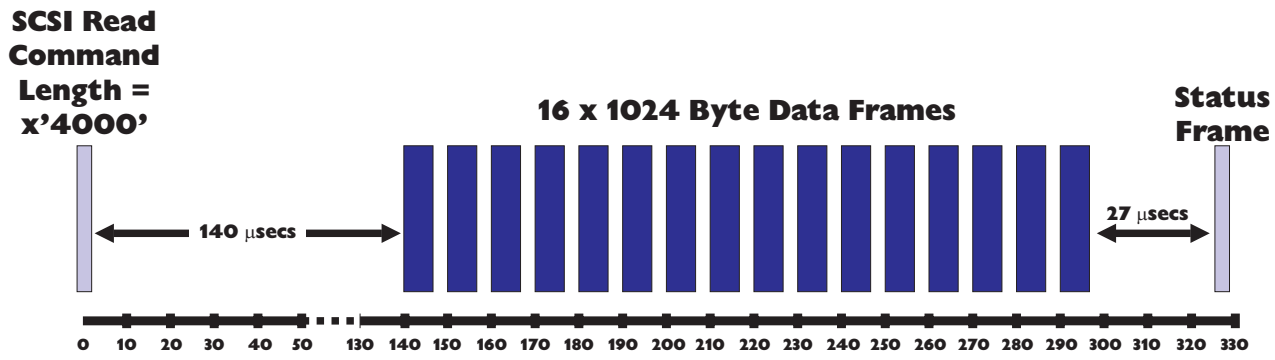


Figure 4. Three phases processing a 16 KB SCSI Read command over a 100 MB/sec FC link.

delay of about 27 µsecs between the last data frame and the transmission of the status frame.

SCSI Write command processing on Fibre Channel runs slightly slower. There is a similar 140 µsec processing delay for the initial SCSI Write command request. As required by the Fibre Channel specification, the protocol requires that the device return an additional setup frame that acknowledges the Write request. In this example, the SDS software impersonating the device generates this setup frame in reply. Following this frame, there is a further delay of 30 µsecs at the client before the data to be transferred from the initiator to the target device appears on the link. In the protocol trace, 1 KB data frames then start to appear at regular 10.275 µsec intervals. After the last data frame, there is a final 30 µsec delay until the SANsymphony SDS issues the SCSI status frame indicating successful completion of the request. Note that the SDS posts a successful completion status following receipt of the full data payload. This is known as a *fast write* to cache. Writes require scheduling an I/O to the disk immediately to update the permanent copy stored there. SANsymphony software schedules the disk update to occur immediately, but it is performed asynchronously so that the SAN client is not delayed for the duration of a physical disk operation.

Overall, processing the Write request takes about 30 µsecs longer than the simpler 16 KB Read request, as illustrated in Figure 5. When the virtual volume is protected using SANsymphony's high availability, disk mirroring feature, Write commands are subject to an additional delay when they are propagated to a standby SDS. When virtual disks are protected by mirroring, successful completion status for Writes is not presented until the write data is present in two SDS caches. This approximately doubles the latency for Write requests.

These timing illustrate the clear performance advantage that FC holds over the older SCSI parallel bus interface technology. A 16 KB Read or Write command can be processed on a Fibre Channel link in roughly the time it takes for SCSI bus arbitration to occur. SCSI bus arbitration and clean-up phases block all other traffic on the bus for the duration of the lengthy signaling and synchronization process. In contrast, because FC bit streams are packet-oriented, it is possible to interleave the packets from multiple commands on an FC link. For example, during the time that one initiator is idle waiting for a reply, another initiator can be transferring data. The capability to overlap processing of multiple SCSI command leads to very effective utilization of Fibre Channel links. On a single 100 MB/sec link, for example, using multiple initiators, it is no problem sustaining 80 MB/sec or better effective data transfer rates. In contrast, effective utilization of a SCSI parallel bus reaches only about 50% of the rated bus speed.

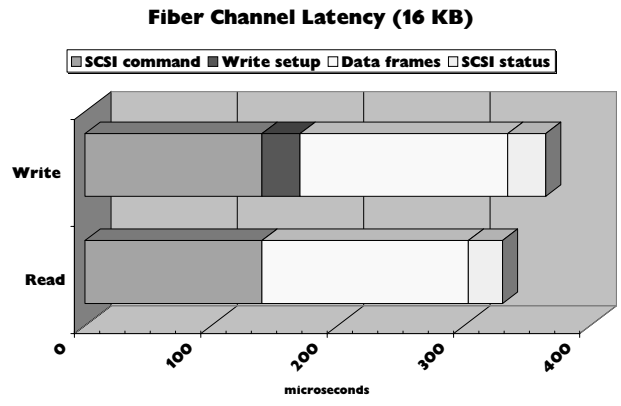


FIGURE 5. A COMPARISON OF THE LATENCY IN PROCESSING 16 KB SCSI READ AND WRITE COMMANDS OVER A 100 MB/SEC FIBRE CHANNEL LINK.

The performance characteristics of high bandwidth, low latency Fibre Channel links, along with the speed and capacity of FC switches, simplifies at least one aspect of SAN capacity planning. Assuming for a moment that the cost per port for an FC switch is not an obstacle, a SAN infrastructure linking hosts to storage devices with no apparent performance constraints can be constructed without much effort. Cascaded switches and long haul connections are complicating factors, as noted earlier, but we suspect these are relatively minor concerns in connecting most server farms to pooled disk resources today.

In practice, of course, it is not cost effective to dedicate an FC link to every host-disk connection. This leads to the usual channel configuration decisions and trade-offs involving string length and string balancing. Native disk device transfer rates remain less than 1/2 the capacity of the FC link. This means that it only takes two or three busy devices (e.g., 3 disks being read in parallel by a multithreaded back-up process) on an FC link to saturate it. So, while it is possible to string together 127 devices on a single FC-AL loop, for example, performance considerations often dictates shorter string lengths. Avoiding I/O hot spots within a complex arrangement of disk and tape devices is also problematic. Until software can be devised to automate the process, SAN administrators will need to rebalance the I/O workload manually across the available hardware periodically.

SAN data management protocols.

SAN data management protocols mediate host computer access to pooled disk and tape storage resources. It has become commonplace to differentiate between *in-band* and *out-of-band* data management. In-band refers to an approach where the data management protocol actively manipulates each and every storage access operation. Out-of-band refers to an approach where the SAN data management protocol is actively involved only in setting up sessions, and is passive thereafter, unless there is an event that changes the state of an active session. One useful way to look at the distinction between in-band and

out-of-band data management is to compare them based on where the data management protocol responsible for virtual to physical disk translation runs during an I/O operation. In out-of-band solutions, virtual:physical disk translation runs on the client. In in-band solutions, it runs on the SAN appliance, transparent to the SAN client.

Out-of-band data management. The most common out-of-band approach requires that the SAN client access a SAN metadata controller (SMDC)[2] during session setup prior to performing any I/O operations to SAN-managed storage resources. Volume and/or file metadata in this context refers to the data about disk volumes and files that a SAN manager component must maintain: e.g., how the virtual volume is mapped to physical disk resources, which SAN clients are authorized to access which volumes and files, what active sessions hold which resources, etc. Once the initial request has been authenticated by the SMDC, the appropriate volume and/or file access metadata is returned to the client, allowing the SAN client to then access managed storage devices directly.

Depending on the approach, having acquired an access token, the client accessing SAN storage proceeds at the either the physical block or logical file level. Again these approaches are distinguished by where on the client the logical:physical translation function is performed. File level accesses require software inserted into the file system layer of the host computer's I/O manager software stack. Block level access requires that the translation occur in the disk driver or Host bus adapter software that interfaces directly with the hardware.

Figure 6 illustrates the sequence of events that structure an SMDC client access session. The session begins when the client requests access to a pooled storage resource (usually a logical volume or file system). The client then waits while it has its request authenticated by the metadata controller. The metadata controller eventually furnishes the SAN client with a token that is required to access the disk or disks involved. This token includes the Fibre Channel and SCSI addressing information that the SAN client needs to access the resource directly. Once its request has been authenticated, the SAN client proceeds to access the storage hardware involved directly, as illustrated.

The performance implications of out-of-band data management protocols are two-fold. First, there is latency associated with accessing the metadata controller during session setup. The extent of this delay is highly dependent on the specific architecture of the SMDC, the contention for this machine in the configuration, etc. But, whatever the specific proprietary implementation, session set-up handshaking is only performed once at session initialization. Even if the SMDC requires periodic reauthorization

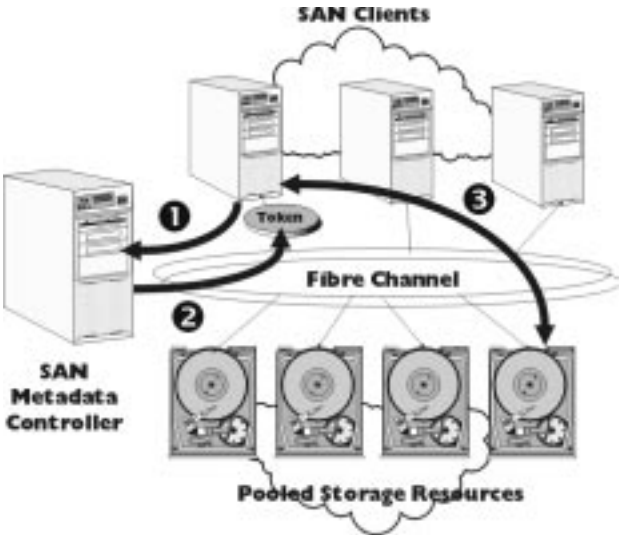


FIGURE 6. THREE PHASES IN SETTING UP A DATA ACCESS SESSION WITH A SAN METADATA CONTROLLER. THE INITIAL REQUEST FOR ACCESS (1) IS AUTHENTICATED BY THE SMDC, WHICH RETURNS AN ACCESS TOKEN TO THE SAN CLIENT (2). WITH THE TOKEN IN HAND, THE SAN CLIENT CAN BEGIN TO ACCESS THE DISK DEVICE DIRECTLY (3).

and renewal of the client's authorization, the performance impact of session initialization should still be relatively minor. So long as SMDC access remains relatively infrequent, the performance impact of this out-of-band access is minimal. This is not the end of the story, however.

Having obtained the appropriate credentials from the SMDC, the SAN client is able to then issue SCSI commands directly to SAN disk and tape devices. The metadata acquired during session initialization provides the SAN client with the information it needs to access SAN storage resources directly from that point forward. To accomplish this, the individual SAN client executes a function that translates virtual device addresses to physical devices addresses. So long as SAN virtual address translation runs as a software function executing somewhere on the SAN client, the performance impact of the SMDC approach should remain low, adding just a few microseconds to device service time. While it is not true that there is no incremental overhead associated with the SMDC approach, it is safe to say that the added overhead per I/O is minimal in most cases. Academic and industry experts that champion this approach to SAN data management make much of this fact.[3] However, this presumed performance edge has not proven to be the overriding success factor that its champions have predicted. In fact, the cost and administrative burden of maintaining a SAN managed by a metadata controller trumps the performance considerations almost every time.

[2] Here and elsewhere we adhere to the SAN classification scheme and other terminology attributed to storage industry analyst Randy Kerns of the Evaluator Group, www.evaluatorgroup.com.

[3] See, e.g., [7]. Dr. Gibson first proposed an architecture for SAN metadata controllers in [8]. For more information on the Gibson's influential approach, see http://www.pdl.cs.cmu.edu/NASD/.

In the SAN metadata controller approach, the price for minimizing the performance impact is the cost of maintaining client-side software for each and every SAN client. When the SAN is used to interconnect only a few computer hosts, maintaining SAN data management software on each client may not be prohibitive. But as the number of SAN clients increases and the SAN is used to interconnect hosts running different operating systems, the requirement to distribute and maintain software on each SAN client grows more burdensome and expensive. The software installation and maintenance burden associated with SAN metadata controllers has slowed down their rate of adoption considerably.

In-band data management. The leading alternative to the out-of-band SAN metadata controller approach is in-band SAN data management. The in-band approach was initially dismissed by many industry observers as somehow too intrusive and technically inferior to the "direct" device access method offered using an SMDC. However, in-band data management protocols do not require SAN client-side software, so they are much easier to deploy. Once deployed, they are also easier to grow and modify. No client software to purchase makes in-band SANs less expensive to acquire and dramatically less expensive to administer. In the hands of capable practitioners, in-band SAN data management has emerged as the more flexible, more cost-effective, and more easily deployed approach. An in-band implementation can also outperform an SMDC architecture, which is the topic we intend to explore at some length here.

SAN appliances. There are several ways to implement an in-band SAN data management protocol. One of the most popular is to use a SAN storage appliance, a dedicated SAN storage controller of some sort. It has been relatively straightforward for vendors of storage processors like the EMC Symmetrix, IBM Enterprise Storage Server (ESS), or the Xiotech Magnitude to equip these units with front-end Fibre Channel interfaces and turn them into serviceable SAN appliances. In this form of SAN appliance, the necessary SAN data management services such as virtualization run inside the appliance, along with more traditional storage data management services like caching and RAID mapping. Running virtual:physical disk translation inside the appliance makes access to SAN storage resources transparent to SAN clients. Since changes applied at the appliance are immediately visible at clients, there is also the benefit of centralized administration. However, this benefit tends to diminish as more appliances are added to the SAN or appliances from different vendors are deployed.

Before proceeding to a discussion of the performance aspects of in-band SAN appliances, it is appropriate to address a common point of confusion about them. It may not be immediately obvious that a storage processor outfitted with front-end FC interfaces functions as an "in-band" SAN storage manager. However, storage processors have long relied on in-band data management protocols, redirecting logical volume requests to RAID-mapped disk arrays, for example. Specifically, a standard storage processor function allows a storage administrator to group physical devices, specify a RAID mapping (RAID 0/1 disk striping with disk mirroring, for example, or RAID 5 redundancy with a rotated parity disk), and export that grouping as a SCSI LUN (Logical Unit Number) that is directly addressable by a host computer. Each host computer-generated SCSI command to read or write some logical disk address must be processed in-band by the storage processor and translated according to the RAID mapping scheme into an actual physical disk location. For example, a direct-attached host computer sees a SCSI LUN that represents a 72 GB physical disk. The storage processor understands that the 72 GB LUN is a logical entity formed by creating a RAID 5 4+1 parity group using five 18 GB capacity drives. Logical:physical disk translation, which is transparent to the attached host, is performed by a control program inside the storage processor that figures out which location on the five physical disks associated with the LUN to Read or Write.

The same, effective techniques for transforming host SCSI commands carry over to current SAN appliance products. Perhaps, due to all the public fuss about emerging SAN technology, people expect more from a SAN appliance than a simple retooling of conventional storage processors to support Fibre Channel interfaces. But, a quick survey of SAN appliances available from the major hardware vendors shows little architectural difference between current products and the previous generation of storage processor equipment that could only speak SCSI.

The SAN data management protocol being incorporated into every I/O operation is, on the face of it, intrusive. With in-band protocols, a SAN data controller processes every individual client access request involving a SAN-managed storage device. At a minimum, for example, logical:physical disk translation occurs in-band. This proven, effective approach is very appropriate for SAN appliances to adopt. Even though in-band redirection of SCSI commands is intrusive, it is performed routinely by all enterprise-class storage processors. This intrusion is justified on several grounds, not the least of which is necessity, allowing storage processors to implement a variety of useful data management services. In the case of RAID, for example, in-band transformation of host SCSI commands to support RAID provides extended disk reliability. Necessity being the mother of invention, in-band intrusion is further justified when one of the services storage processors routinely provide is caching frequently accessed data in RAM to improve performance.

As noted above, there are only relatively minor architectural differences between previous generation storage processors and current SAN appliances from the same vendor. What architectural changes that have occurred were motivated by performance considerations raised by high bandwidth, low latency Fibre Channel technology. The major engineering challenge storage processor vendors face in transforming older designs into SAN appliances is ensuring there is sufficient internal processor speed and internal bus capacity to handle high-speed FC interfaces. Storage processors that were originally built using a number of small, inexpensive microprocessors (e.g., the Intel i960) designed specifically for real-time applications may need a power boost when they encounter FC bit streams. Internal data bandwidth can also be an issue. A recent example is the EMC CLARiiON FC4700 that utilizes twin Intel Pentium III 733 MHz microprocessors, much faster hardware than the previous generation CLARiiONs that used PowerPC chips.

The fact that there are only minor differences between current SAN appliances and previous generation SCSI-attached storage processors simplifies capacity planning. The newer machines perform similar to the older ones, except that they benefit from faster front-end and internal FC interfaces. As noted earlier, the FC protocol eliminates the time consuming SCSI bus arbitration phase. Because it supports packet interleaving, it can sustain effective data transfers rates that are fully 80% of the total available bandwidth. Finally, it provides faster data transfer rates clocked at 1 Gb/sec.

A general rule for projecting the performance improvement of these subsystems is that so long as internal processor speed and bus capacity are not a constraint, storage processor performance scales according to front-end capacity (i.e., FC interface speed) for cache-friendly workloads and according to disk speed for write-oriented and cache unfriendly workloads. If you know the throughput capacity of storage processor using slower SCSI or ESCON front-end interfaces, it is straightforward to project the performance of the same box retrofitted with faster FC interfaces by factoring in the improvements discussed in Figures 4 and 5 above.

SAN data managers. A second way to implement an in-band SAN data management protocol is to incorporate SAN data management functions into an existing computing platform. This approach, which has become known as a SAN storage manager, is distinguished from a SAN appliance by the fact that it is not an extension of an existing disk storage subsystem. A SAN storage manager is a separate, dedicated component placed between interconnected host computers and disk and tape peripherals. Below we will have the opportunity to look in detail at the performance of one SAN storage manager, Datacore Software's SANsymphony.

Datacore's alternative to building a SAN storage manger is a software-only approach that utilizes off-the-shelf Intel server hardware running a standard version of the Microsoft Windows 2000 operating system attached to any variety of standard disk and tape peripherals. Minimum requirements for SANsymphony are an Intel server with two processors running Windows NT or 2000, at least one FC Host Bus adaptor, and 256 MB of RAM, normally configured as a standalone storage management machine. In addition, a cluster of at least two SDS PCs are normally configured for fault tolerance and continuous operations. With Datacore's SANsymphony software, it is possible to create a fully functional SAN storage manager using common, inexpensive hardware components, including Intel-based servers, Windows 2000-compatible disk and tape subsystems, and FC Host Bus adaptor (HBA) interface cards.

SANsymphony Storage Domain Servers (SDSes) implement an in-band SAN data management protocol, as illustrated in Figure 7. All client access to pooled disk resources is routed through the SDSes. Similar to a SAN appliance, virtual:physical disk translation occurs inside the SDS. Data access requests are then either satisfied from internal server RAM configured as a cache or an I/O to disk scheduled by the SDS directly. (Functionally, this is quite similar to what goes on inside a SAN appliance.) As indicated, multiple SDSes can be configured into a N+1 fault tolerant cluster. An SDS cluster provides fault-tolerant disk mirroring with automatic failover to a surviving node or mirrored disk.

Unlike conventional storage processors that are available in a limited variety of standard configurations, a SANsymphony Storage Domain Server (SDS) can be assembled from a wide range of equipment alternatives. In fact, the sheer number of hardware alternatives to consider can be daunting, ranging from simple 2-way Intel servers to 4 and 8-way machines with 4 GB or more of RAM. Besides sizing the server machine where SANsymphony runs, the capacity planner must also decide (1) how to configure the attached disk subsystem – the amount of disk capacity,

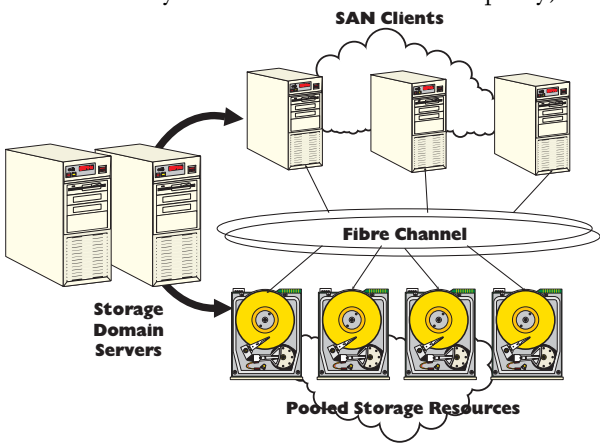


FIGURE 7. AN IN-BAND SANSYMPHONY STORAGE DOMAIN SERVER.

number of disk actuators (devices), RAID mapping, string depth, internal pathing, etc. – and (2) how to configure the FC switched fabric.

Below, we will discuss sizing a SANsymphony Intel server, focusing specifically on the areas of processor speed, number of processors, and PCI bus bandwidth.

The I/O capacity of an SDS

Because the SANsymphony SAN storage manager software runs on a standard Intel-based server, the performance and sizing considerations for an SDS are identical to the ones that arise with standard Windows 2000 Intel application servers. This allows us to make use of standard performance and tuning tools that operate in that environment. We can also draw on existing expertise in Windows 2000 performance topics.[4] For SANsymphony Storage Domain Server sizing, this general knowledge of Intel server hardware and Windows 2000 operating system performance needs to be augmented by specific expertise in the operation of the SANsymphony software.

For the sake of simplicity, in this document we will assume that all FC connections between SAN clients, SAN-managed disks, and SANsymphony SDS machines can always be made using a single Fibre Channel switch to eliminate from consideration the complexity cascading fabric switches introduces. Given the availability of switching gear capable of packaging as many as 256 ports in a single unit, this appears to be a reasonable simplifying assumption.

Rather than discuss SANsymphony performance in the abstract, we will illustrate the discussion with measurement data obtained from exercising an SDS in a benchmarking test environment. We conducted a series of benchmarks, only some of which space permits describing below, in an attempt to characterize SANsymphony performance over a broad range of conditions. We will try to place the measurements we obtained into a conceptual framework that allows you to generalize from these specific results.

Benchmark environment. We obtained all the measurements discussed here from a single SDS node running a prototype of SANsymphony version 5. The SDS machine was a 4-way 550 MHz Intel server running Windows 2000 with Service Pack 1 installed. The server contained three separate PCI buses, all running at 33 MHz. Two of the buses supported 32-bit cards, while the third could handle PCI-2 64-bit cards. The SDS contained 9 QLogic 2200 FC adaptors. The configuration of this SDS machine is depicted in Figure 8. Seven Intel

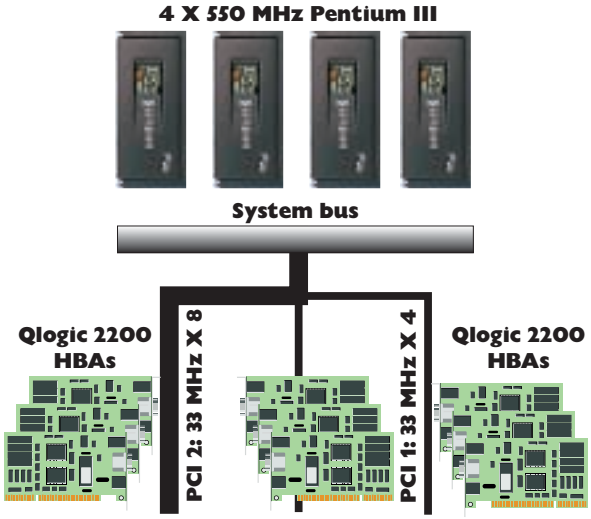


FIGURE 8. THE CONFIGURATION OF THE SDS SERVER USED IN THE BENCHMARK TESTS.

clients, all running Windows 2000 on 2-way 300 MHz workstations, were attached to the SAN. Each client contained a single QLogic 2200.

Two racks of 10 Eurologic SANbloc disks featuring Seagate 10,000 RPM Cheetah FC drives were used to create SAN client disks. The SANsymphony SDS server, Intel clients, and JBOD disks were all interconnected using a 32-port inRange fabric switch. Using SANsymphony's SANManager administrative tool, we created 14 virtual volumes, each on a dedicated physical disk, and assigned them two at a time to the SAN clients. We installed the Intel Iometer disk benchmarking utility on one of the client machines and the remote Dynamo benchmarking agent on the remaining six clients. The SDS machine and the SAN clients were all instrumented using Performance SeNTry from Demand Technology Software, performing data collection every 10 seconds. The Windows 2000 defaults for **`diskperf`** were used, collecting performance statistics for Physical Disk data only. Besides disk performance, we also collected data at the system, process, and networking level.

In addition, we collected SANsymphony internal performance statistics, which are available as standard Windows 2000 performance Objects and Counters. SANsymphony provides extensive measurements on SDS throughput, cache effectiveness, and other areas of interest. Understanding the SANsymphony performance statistics requires some knowledge of the internal architecture of the software, a topic we explore below.

SDS software architecture. A SANsymphony Storage Domain Server is a conventional Intel server running Windows 2000 and the SANsymphony software package that allows the server to function as a dedicated in-band SAN storage manager. The SDS software architecture is conventionally represented as a stack where each layer

[4] Microsoft includes a performance and tuning applet called System Monitor with every version of the Windows 2000 operating system. Additional Windows 2000 performance tools and documentation are available in the Windows 2000 Resource Kit. [9] is an excellent general purpose technical references on Windows 2000 OS internals. Specific technical references for Windows 2000 server performance, tuning, and capacity planning topics are [10] and [11].

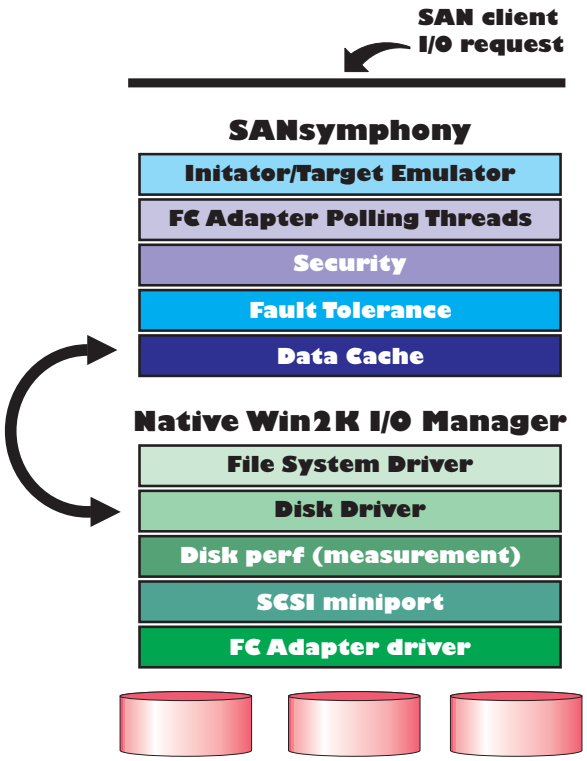


FIGURE 9. THE LAYERED ARCHITECTURE OF THE SANSYMPHONY SOFTWARE STACK.

optionally operates in sequence on SAN client I/O requests, as depicted in Figure 9.

Dedicated kernel threads with operating system level privileges run the stack. They begin by interrogating the Fibre Channel interface cards installed in the SDS, capturing individual SAN client I/O requests, and operating on them. To achieve higher levels of performance than is otherwise possible on an all-purpose computing platform like Windows 2000, SANsymphony uses polling, where its kernel threads run continuously interrogating the Fibre Channel Host bus adaptor interface cards, constantly looking for work. An FC HBA polling thread is significantly more responsive than an interrupt-driven mechanism that is better behaved in a general purpose computing environment. Because polling provides more predictable device service time, it is the technique used in most real-time environments, including dedicated communications and storage controllers. Of course, polling is less efficient so it is not the approach used on servers designed for general purpose computing. Users are cautioned to run SANsymphony on a machine that is dedicated to running the SDS, which is a commonsense way to deploy the software anyway.

Even though it uses polling, the SANsymphony software is designed to co-exist reasonably well with other applications that might be running on the same Windows 2000 server. In practice, these other applications are likely to be limited to things like the SANsymphony GUI, performance and storage resource monitoring, and native disk-to-tape backup utilities. SANsymphony creates a maximum of *n-1* polling threads where *n* is the number of processors available for use so there is always some residual capacity available for other applications. At no point will SANsymphony ever take over all the processors in a standard Win2K symmetric multiprocessing configuration, always leaving at least one processor available for assorted Windows 2000 housekeeping functions. Using an adaptive mechanism that recognizes when client I/O requests are likely to begin queuing, SANsymphony starts with one dedicated polling thread, but will then automatically dispatch additional polling threads as the I/O workload increases. At any one time, the number of SANsymphony kernel threads that are dispatched varies based on the load. In case the remaining Win2K non-Fibre Channel initiated processing load is constricted on a single processor, the maximum number of SANsymphony polling threads can also be configured manually.

Whenever an SDS polling thread determines that an FC interface card has a request outstanding, it is processed in-line by the kernel thread. As indicated in the diagram, the FC polling threads fields both initiator and target SCSI device requests. SAN clients issue I/O requests to virtual disk targets, which are indistinguishable from ordinary SCSI physical disks. These requests are intercepted and interpreted by SANsymphony's Target Mode emulator software, which impersonates the physical disk entity served up by the SDS. Executing the next layer of the SANsymphony software stack, the thread then authenticates a request to Read or Write a SANsymphony virtual disk. If valid, the request is processed by the memory caching layer. If the data requested is already resident in memory, it is a *cache hit* and the client I/O request is processed immediately.

On a Read cache miss, the SANsymphony kernel thread creates a standard I/O Request Packet (IRP) that is passed to the native Windows 2000 I/O Manager to read data from the associated physical disk. This allows SANsymphony to manage any disk device that can be directly attached to the SDS, including native SCSI disks. The data requested is copied first into cache memory and then returned to the SAN client across the Fibre Channel link.

On a Write, the request is passed through to a fault tolerant layer that supports both synchronous and asynchronous disk mirroring using an active link to another node in the SDS cluster. The fault tolerant layer is responsible for keeping the mirrored SDS node and disk in sync with the original, so it schedules an I/O to the other SDS. Once a copy of the data is safely resident in two SDS caches, the Target mode driver signals the SAN client that the Write request was successful. Meanwhile, both SDS nodes schedule a physical disk update to harden the changed data on auxiliary storage as soon as possible.

Processor consumption. First, let's look at processor consumption for various SANsymphony functions on an SDS node. We will look at both best case performance when the SDS has little to do except poll its Fibre Channel interface cards looking for work and worst case performance when the SDS is serving up small Read I/O requests from cache. The latter operation, which leads to the highest I/O rates, is the most CPU intensive of any SANsymphony function.

When SAN clients are idle and the Fibre Channel interface polling engines on the benchmark machine have nothing to do except look for work, SANsymphony running on our benchmark machine recorded slightly more than 375,000 unproductive polls/sec. That works out to one kernel thread checking a Fibre Channel interface card every 2.66 µsecs. Except for the activity associated with the polling thread, the system was otherwise idle. The overall CPU utilization for the system was just over 25%, reflecting the relentless polling activity and very little else (except for active measurement tasks). This 2.66 µsecs of delay associated with an unproductive device polling sequence represents the minimum latency of a FC interface request.

Now let's look at the timing of a successful polling operation. With seven SAN clients issuing small block 512-byte requests that could be satisfied from cache, SANsymphony increased the number of polling threads to the maximum level of three on the 4-way 550 MHz test machine. Together, with three polling threads active, the SDS machine was able to process about 40,000 I/O requests per second. At 40,000 IOPS, SANsymphony recorded 80,000 productive polls. Each I/O request processed by the SDS evidently required two successful FC interface polls – one to accept the client's original Read request and a second after the data was transferred from cache memory back across the FC link to the client.

In addition to the productive polls, the polling engines still managed to issue more than 31,000 unproductive

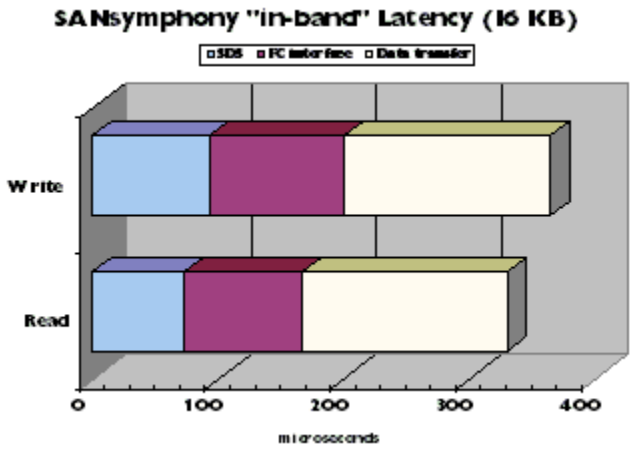


FIGURE 10. SDS IN-BAND LATENCY AS A FUNCTION OF OVERALL SERVICE TIME FOR A 16 KB READ.

polls in the midst of all that real work. Assuming the same 2.66 µsecs to complete an unproductive polling operation, the unsuccessful polls consumed about 83 ms. of processor time each second. With three processors running full tilt at 100% busy (the fourth was almost idle, except for measurement tasks), we can then calculate the latency associated with an in-band operation:

((3,000,000-(31,000*2.66))/40,000

namely, 73 µsecs of processing delay per request.

Notice that this 73 µsecs in processing time occurs over two successful polls. The first set of processing that the SDS performs occurs during the 140 µsec delay illustrated in Figure 4 following the transmission of the SCSI Read command. The second successful poll occurs after the SDS sends the last data frame, prior to sending the status frame. SDS in-band processing represents 73 µsecs during two protocol operations that together last about 167 µsecs. We conclude that "in-band" SDS processing delays account for about 44% of the service time, the remainder apparently taking place inside the interface card itself, which contains a processing element considerably less powerful than an Intel Pentium III.[5]

During the Write operation illustrated in Figure 5, there is a third successful poll issued to return the Write setup acknowledgement frame. We note that the delays associated with both the Write acknowledgement and the SCSI status command account for about 30 µsecs of latency. The simple processing associated with handling these events in both the SDS polling thread and the FC interface card suggests some minimum latency factor for in-band transactions in this hardware environment, in this case about 30 µsecs. Speculating about how to apportion this minimum latency between the SDS internal code path and the interaction with the FC HBA card leads to a picture like Figure 10, detailing the relative contribution of each processing component for both Reads and Writes.

Capacity planning. We were able to measure indirectly approximately 73 µsecs of delay that takes place inside the 550 MHz Intel server benchmark system associated with SANsymphony in-band processing of a SCSI disk Read command. It should be evident from this measurement that the SDS code stack is not a very CPU-bound process. In fact, the only way to simulate a processor bottleneck in our test configuration was with an artificial workload using uncharacteristically small 512-byte blocks. For more realistically sized 4K and 16K blocks, the SDS configura-

[5] Processor capacity inside the HBA has a major impact on performance, something that is evident when you test successive generations of the vendors' cards. In the case of the QLogic 2x00 series, for example, we saw an effective throughput increase of 40-60% in moving from the 2100 to the 2200, with a similar boost provided by current generation 2300 cards. For this test configuration we did not have enough of the newer 2300s to go around, so we opted to use the 2200s throughout to maintain uniformity.

tion bottleneck quickly shifts to other resources, as we will discuss in a moment.

It is worth noting that the SDS internal processing delay should scale with the speed of the processor on the SDS server. On an Intel server equipped with 1 GHz engines, for example, we expect that the in-band processing delay would drop below 40 µsecs. On today's top of the line 1.7 GHz processors, latency falls further to below 24 µsecs per I/O operation. At these faster processing speeds, it becomes even more important to step up to the latest generation Fibre Channel interface cards to take full advantage of faster CPUs. In Figure 11, we project the latency for a 16 KB Read request for next generation hardware, upgrading first to a 1.7 GHz processor and then upgrading to the 200 MB/sec FC HBAs that are just beginning to appear. We project that SAN-symphony total device service time can be reduced a little over 50% using next generation Intel and FC hardware.

This is also a good place to remark upon some of the internal statistics that the SANsymphony software maintains. One noteworthy measurement to track in the current context is the rate of productive polls, calculated as

$$Productive polls/sec * 100 / (Productive polls/sec + Unproductive polls/sec)$$

Internally, SANsymphony calculates this ratio every few seconds to determine whether sufficient polling threads are active. When the rate of productive polls rises to 50%, SANsymphony will dispatch another polling thread, up to the maximum number of polling threads permitted (one less than the number of processors). As a configuration rule of thumb, if the rate of productive polls remains above 50% and SANsymphony is running the maximum number of polling threads for an extended period of time, there is a CPU capacity constraint. In this benchmark test, for instance, the rate of productive polls was 72%. Configuring additional engines, faster engines, or some combination of both is warranted under these circumstances.

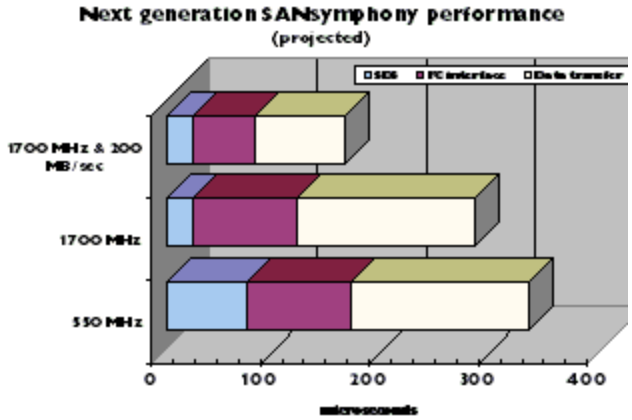


FIGURE 11. PROJECTING SANSYMPHONY PERFORMANCE ON NEXT GENERATION PC AND FC HBA HARDWARE.

SDS scalability. Because the CPU processing load of an SDS in-band SAN storage manager is modest and it is relatively easy to procure fast Intel engines, users of a SANsymphony SDS are not likely to encounter serious processor bottlenecks. Testing with more realistically sized 4 and 16 KB blocks, the potential bottleneck shifts to internal PCI bus bandwidth for cache-friendly Read workloads and to the attached disk subsystem for Writes and very cache-unfriendly workloads. These performance expectations are common to all SAN appliances, as noted earlier. The chart below in Figure 12 highlights these critical considerations. Let's look at the impact of PCI bus bandwidth first.

The PC Server used in this testing was equipped with one 33 MHz 64-bit PCI version 2 bus and two standard 33 MHz 32-bit PCI buses. The server was populated with 9 QLogic 2200 HBAs arranged three deep on each PCI bus, as illustrated back in Figure 8. The results in Figure 12 summarize a series of tests we performed using identical workloads, while varying the number of HBAs (and PCI buses) involved. Results showing both maximum I/O rates (achieved using 512-byte blocks) and throughput (using 16 KB blocks) are shown.

In the first test series, the SDS server was set to use only one FC HBA for Target mode requests – remember, Target mode refers to SAN client requests to access SANsymphony virtual disks. This HBA was plugged into one of the available 32-bit buses, by the way. A second HBA was enabled to provide connectivity to the Eurologic FC disk arrays. Because we were not interested in spotlighting any disk subsystem performance issues at this time, we maintained a single FC back-end connection throughout all tests. We should note that the FC link to the disks was dormant anyway during this series of tests because we artificially restricted the workload to fit in the server's memory-resident cache. There was almost no need to access the back-end disks.

This first scenario simulates running the SANsymphony storage manager software on a low cost Intel server equipped with a single PCI bus. This basic configuration reached almost 78 MB/sec throughput using 16 KB blocks. It reached a maximum I/O rate near 6800 I/Os per second with 512-byte blocks.

In the second series of tests, we enabled a second HBA on the first 32-bit PCI bus and added two HBAs that were plugged into the faster 64-bit slot (for a total of four front-end interfaces). This scenario simulates running SANsymphony on a higher-priced dual PCI-bus server. Note the impact of doubling the PCI bandwidth and the effect of doubling up on the number of HBAs configured per bus. Both the I/O rate and maximum throughput almost tripled to 18,000 I/Os per second and 210 MB/sec throughput.

In the third scenario, we enabled all eight available HBAs for SAN client connections, with two configured

on the 64-bit bus and three on both 32-bit buses. At this point the configuration on this high end, triple PCI bus PC server was maxed out. The HBA connected to the back-end disks occupied the third slot on the 64-bit bus, while a NIC was installed in the only other available slot on the motherboard. Doubling the number of front-end FC connections and utilizing the third PCI slot led to another doubling of the maximum I/O rate to 40,000 IOPS, which was the CPU-bound scenario we discussed at length above. Maximum throughput of this configuration also increased, but only to about 270 MB/sec. The bottleneck in this latter 16 KB scenario is not immediately apparent, but is evidently not the result of insufficient server processing power, as we shall see.

Figure 13 illustrates the adaptive algorithm that SANsymphony uses to dispatch polling threads. At the beginning of the benchmark run, there is one polling thread dispatched. The rate of productive polls is near 0%, and the 4-way system idles along at just above 100% total CPU busy. Notice that the kernel thread is well behaved, allowing the OS to schedule it to run on any available processor.

When the benchmark 16KB I/O workload kicks in shortly before 18:01, the productive polling rate rises sharply to 50%. But then SANsymphony releases a second polling thread, pulling down the productive polling rate to 25% for the remainder of the test. With two polling threads active, SANsymphony boosts CPU utilization to 200%. Since the number of active polling threads reaches three (and CPU utilization remains at 200%), it is evident that the configuration is not CPU-constrained for this workload.

Unfortunately, the PCI bus on Windows 2000 servers is not instrumented, so we could not measure bus utilization directly. However, based on its known operating characteristics, we can project that the bus configuration was saturated during this test. Since the FC HBAs connected to the bus can only transfer data across the bus at a rate of 100 MB/sec, there is a corresponding reduction in the effective bandwidth of the PCI bus to which they are attached. Even the faster 33 MHz eight-byte wide version 2.1 PCI bus can only sustain about 90 MB/sec throughput because the interface cards cannot run any faster. Upgrading to 200 MB/sec FC HBAs cards looks like a promising way to reach higher throughput levels. However, 200 MB/sec cards can only run at that speed when plugged into a PCI 2.1 slot. Moving to 200 MB/sec HBAs also triggers a succession of hardware upgrades to the SAN clients and to the FC switch that is used to

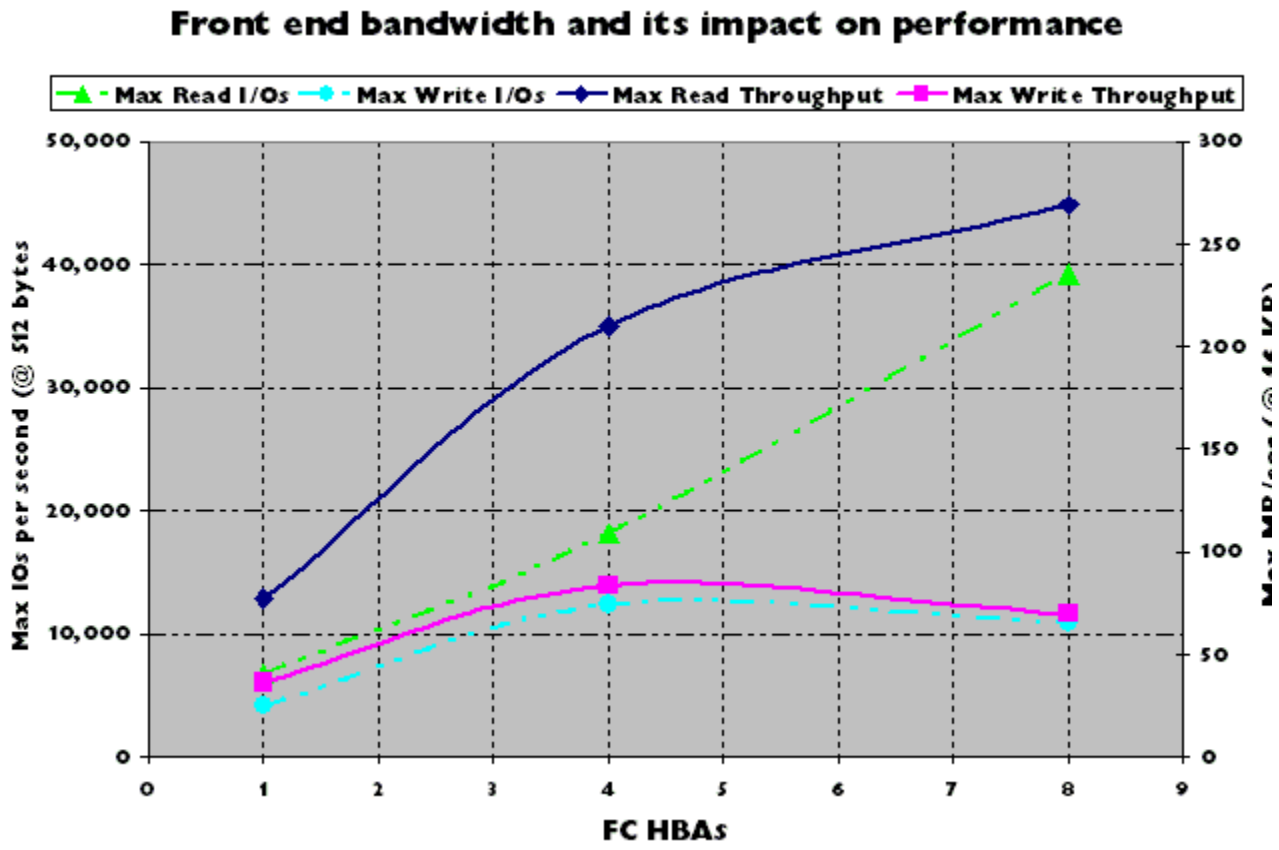
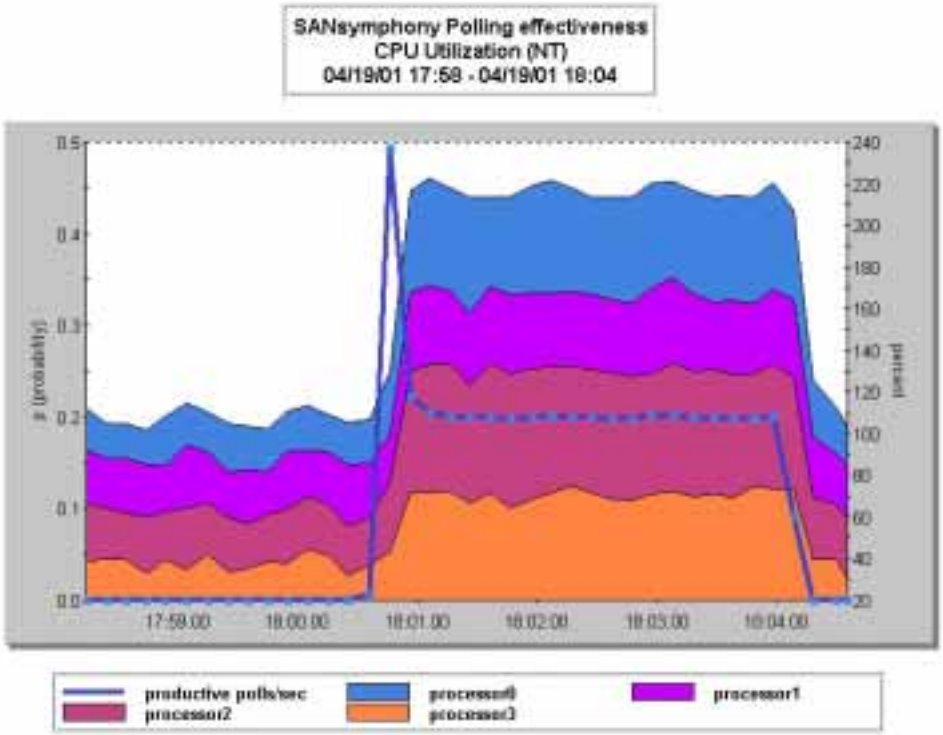


FIGURE 12. FRONT END BANDWIDTH AND ITS IMPACT ON SCALABILITY.



interconnect them before higher throughput rates can be attained in any configuration.

Conclusion.

This paper discusses basic capacity planning considerations for popular forms of in-band and out-of-band SAN data management protocols. We were able to present and discuss performance data from a series of benchmark tests designed to clarify configuration and tuning issues involving SANsymphony Storage Domain Servers, an in-band SAN storage manager. Because SANsymphony software runs on standard Intel server hardware running the Microsoft Windows 2000 operating system, its performance issues are similar to other Intel server workloads. The availability of professional quality performance measurement tools for the Windows 2000 platform, along with the expertise in knowing how to use them, makes it possible to explore these performance issues in some detail.

For a typical 16 KB block Read satisfied from SANsymphony memory-resident cache, we measured a 330 µsec device service time, broken down into three distinct command processing phases associated with the Fibre Channel serial protocol. Of the total 330 µsecs of elapsed SCSI target device service time, about half involved transfer of the 16 KB data payload, while the other half was due to processing by the FC interface card and the SDS software stack. For the 4-way 550 MHz PC used for this series of tests, SDS software accounted for just 73 µsec out of the total 330 µsec latency that was recorded. The in-band delay experienced by SAN clients accessing a virtual disk powered by a moderate cost SANsymphony appliance compares favorably with the fastest (and most expensive) FC-enabled storage processors available.

Since the SDS software stack requires only modest amounts of CPU power, realistic I/O workloads are far from compute-bound on widely available Intel-based servers. As the benchmarks results from tests where we increased the number of FC interface boards and PCI buses used in the sample configuration showed, SANsymphony performance scales in a predictable manner with the number of HBAs and the internal bus bandwidth of the server machine.

References

[1] Brandwajn, Alexandre, "A study of Cached RAID-5 I/O," CMG *Proceedings*, December, 1994.

[2] McNutt, Bruce, *The fractal structure of data reference: applications to memory hierarchy*. Boston: Kluwer Academic Publishers, 2000.

[3] Chung, et. Al., "Windows 2000 Disk I/O Performance," Microsoft Research Technical Report MS-TR-2000-55, available at ftp://ftp.research.microsoft.com/pub/tr/tr-2000-55.pdf.

[4] Evio Valdevit, "Cascading in Fibre Channel: how to build a multi-switch fabric," Brocade Communications Systems, available at http://www.brocade.com/SAN/white_papers/pdf/Cascading.pdf.

[5] John Curtis, "In defense of Jumbo Frames," *NetworkWorld,* August 10, 1998.

[6] Peterson and Davie, *Computer Networks*, San Francisco, CA: Morgan Kaufmann, 1996.

[7] Gibson and van Meter, "Network-attached storage architecture," *Communications* of the ACM, November 2000.

[8] Garth Gibson, "A Case for Network-Attached Secure Disks." Carnegie-Mellon Technical Report, 1996.

[9] Solomon and Russinovich, *Inside Windows 2000,* Redmond, WA: Microsoft Press, 2000.

[10] Friedman and Pentakalos, *Windows 2000 Performance Guide,* Boston, MA: O'Reilly Associates, 2001.

[11] Curt Aubley, *Tuning and Sizing Windows 2000*, Englewood Cliffs, NJ: Prentice-Hall, 2001.