Chapter 7

Connectivity Options

In previous chapters, we discussed how effective storage systems are developed from the integration of multiple storage devices. Understanding the structure of these devices can help effectively utilize these technologies to support a single complex workload or a combination of disparate workloads. Underlying these concepts of storage architecture, device operation, and anatomy are the connectivity strategies that complete the data highway system. All these elements will begin to come together in this chapter as we discuss the connectivity options available to transport and deliver data from the storage system components to the computer.

Traditional system connectivity strategies have long been based upon bus architectures. A computer bus is defined as the lines and controls that connect the diverse components of a computer, ultimately making it a complete system. As we discussed in Chapter 5, this connectivity is required for the internal server components, CPU, RAM, and cache. Obviously, we know that external I/O components, such as storage devices, disks and tapes, and other peripherals like displays, keyboards, and mouse devices also require this connectivity, but in a different manner. This chapter will begin to explain the components and operation of typical bus- and network-oriented technologies, and the creative uses of various buses and network connectivity components that form the foundation for large multiple user systems.

The ability to connect storage systems through networks has greatly impacted the storage I/O industry. All of it surrounds the connectivity strategy, and the movement of storage systems from discrete bus technologies to network-oriented architectures. Understanding the bus is paramount to leveraging the movement of storage I/O into this new network model.

Another important reason to cover traditional bus technologies contrasted against the new network interconnects is that existing buses do not go away. They remain, and will remain, fundamental to getting data to and from the CPU (refer to Chapter 5 for more information). Moving data from the storage device through the storage system and on to the server within a network infrastructure still requires the effective operation of an internal bus. In order to be effective, traditional bus components must be optimized since they become parts of a new storage connectivity infrastructure. Otherwise, the most advanced and feature-rich storage network falls short at the server.

Connections: Getting on the Bus

Internal traffic within a computer's motherboard is handled through a set of fast connections etched into the circuitry, enabling transfer of data to and from the CPU and internal temporary data locations—for example, RAM, cache, and CPU registers. The connectivity for external (external to the motherboard) hardware is handled through the edge (of the board) as an expansion to the board. Take a look at Figure 7-1, which shows a universal computer bus.

These connections to the outside world (external to the motherboard) are referred to as expansion buses and I/O channels. They provide hardware attachment mechanisms

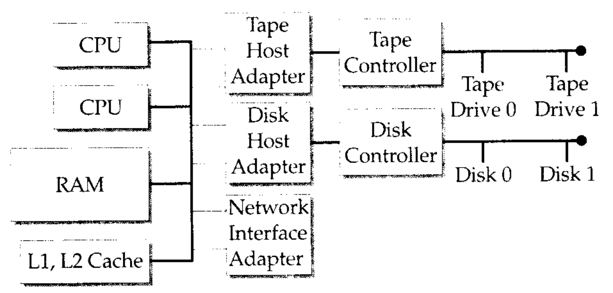

Figure 7-1. *A universal computer bus*

that extend the system capabilities beyond what is on the motherboards. This is where I/O devices are connected and, as we discussed earlier, where the host adapter components are attached. It is here where we require our first transfer ticket provided by the host adapters, to get from one bus to the other. As the internal components require a bus to connect to each other, the I/O devices also have that same need. As Figure 7-1 illustrates, the internal bus, I-bus, is adapted to the I/O bus, each transferring and controlling data for the connected components. Each bus controls devices with different mechanisms, different timings, and capacities.

The buses provide the following fundamental functions. As we can see, these are largely electronic mechanisms that do their work internally through a host of components. As we will explore later, the bus mechanisms must be able to work with particular devices that speak their language and have common electronic traits to move data.

Underlying all these sophisticated and complex electronics are several fundamental elements that make up the characteristics of the I/O bus, as shown in Figure 7-2. These include: throughput (commonly referred to as bandwidth), address space (how wide the bus is), interrupt processing (or can I react to external events and change processing modes?), and electro-mechanical attributes, such as how long can I be before my signal is degraded, and am I a circuit board or a backplane—in other words, will I connect to the motherboard or will the motherboards connect to me.

Bandwidth

Bandwidth is the amount of information the bus can transport during a given time. This is usually measured in MB per second. This is a derivative of clock speed and data width. In other words, a bus using a 5MHz clock speed and a 4 byte data width has a throughput, or bandwidth, of 5 × 4 = 20MB per second. Keep in mind this is what is called aggregate throughput; it is reduced by protocol overhead. We will discuss real speeds in more detail later in the chapter.

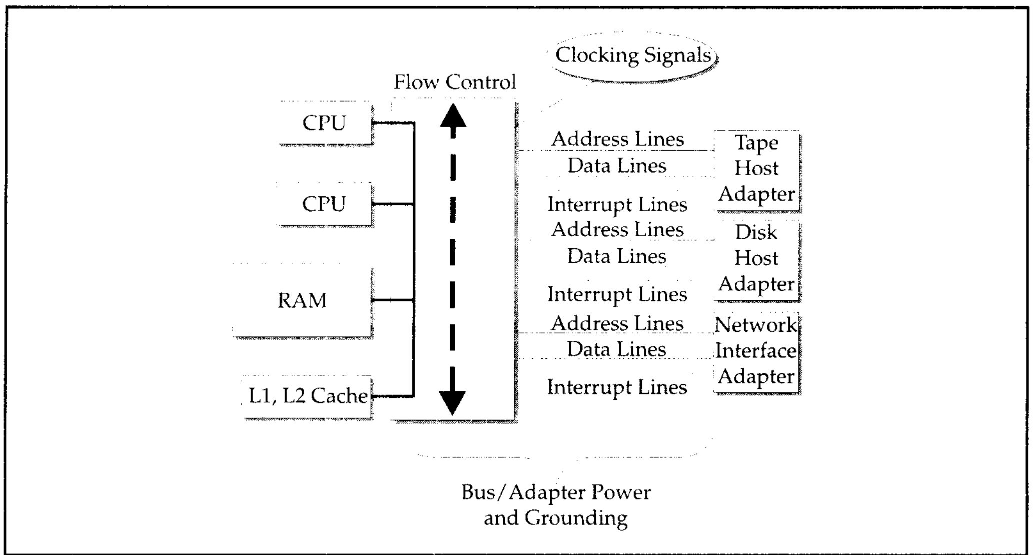

Figure 7-2. *The internal functions of a typical I/O bus*

Addressing

The address space provides the capability to determine unique identities for source and destination of communications within the bus itself. This is determined by the width of the address component; the larger the address space, the more device operations can be supported. Generally, most buses operate today with 32-bit addressing architectures, providing an address space of 4GB. (This is important if you are working with older bus technology or an advanced technology such as InfiniBand or Fibre Channel; see Part V.)

Interrupt Functions

An external event may occur at any time and the bus must be able to take action in order to continue processing. This interrupt processing is the fundamental concept behind the interactive system's processing that must occur with multiuser, multitasking workloads. The I/O bus must be capable of interruption of a data transfer from disk to tape for an event with a higher priority. Be it an error processing routine or a call to read a block of data from an application, without this capability the bus could not be used for interactive applications, instead it could only be used for a single device.

Electro-Mechanical

Internal buses on the motherboard may only be inches long; external I/O connections must extend several feet or meters. These connections are generally some type of shielded cable and will be placed into areas of electromagnetic interference. I/O buses must have attributes that protect the signal's integrity while providing an adequate physical extensibility. The cable for an I/O bus is defined with regard to its maximum length, resistance, whether shielded or unshielded.

The motherboard of a computer will have a number of connections or slots that are nothing more than bus connections. Host adapters are connected at these edge points. The other type of connectivity is the backplane (refer to Chapter 6). The backplane can have no other electronics or circuitry than to provide connection points for host adapters. However, a backplane can also be used to connect complete motherboard systems together along a shared bus. These will be used in tightly coupled SMP and MPP systems. They are also becoming popular for use with integrating thin computing nodes (for example, a system-on-board or a motherboard with no peripherals), which entails sharing a common external bus that may be a network or another type of system interconnect—used explicitly in something referred to as blade computing. (Refer to Chapter 20 regarding future protocols such as InfiniBand, an external example, and Rapid IO, an internal example.)

Bus Evolution

The importance of bus evolution is visible in terms of the clocking signal increases, numbers of data lines, interrupt sharing, and something called bus mastering. Bus mastering allowed additional processors and devices (for example, IDE and SCSI adapters) to contend for bus processing. By contrast, previous architectures where the system processor had complete control of the bus and scalability became problematic.

The decoupling of the bus through bus mastering from a single point of control came with IBM's Micro Channel Architecture (MCA); however, it was enhanced with EISA to include auto and software configuration functions. This first laid the groundwork for plug 'n play functionality, while the second aided in reconfiguring the bus (changing device priorities, addressing, and so on).

Note *This standard bus architecture (Extended Industry Standard Architecture or EISA) extends the ISA standard from a 16-bit to a 32-bit interface.*

The physical decoupling and enhanced scalability of the bus was achieved with the Peripheral Component Interconnect Standard (PCI) and the resulting PCI mezzanine bus architecture. This was the first bus implementation not tightly coupled with the system processor. This allowed the PCI bus to expand its connectivity by using a re-configurable structure and bridging, as necessary, to add additional hardware devices.

Bus Operations

Buses operate by requesting the use of paths to transfer data from a source element to a target element. In its simplest form, a controller cache, defined as a source, transfers data to a disk, defined as a target. The controller signals the bus processor that he needs the bus. When processing, control is given to the controller cache operation. It owns the bus for that operation and no one else connected to the bus can use it. However, as indicated earlier, the bus processor can interrupt the process for a higher priority process and give control to that device.

This is called a bus arbitration scheme, as shown in Figure 7-3. The devices connected to the bus arbitrate for bus control based on a predefined set of commands and priorities. The devices can be both source and target for data transfers. This type of internal communication makes up the bus protocol and must be available to all sources and targets—for example, connected devices that participate within the I/O bus.

Parallel vs. Serial

Bus operations will perform data transfer operations using parallel or serial physical connections. These two connections perform the same task, but have distinct architectures, each with their own pros and cons. It's likely you'll be confronted with each as storage systems begin to integrate both for their strengths. Over and above this option is the secondary choice of the type used within each category, such as point-to-point and differential for parallel SCSI and various serial implementations. It's important to understand the limitations, as well as the strengths, when applying these to particular application requirements.

Parallel connections for storage utilize multiple lines to transport data simultaneously. This requires a complex set of wires making parallel cables quite large and thus

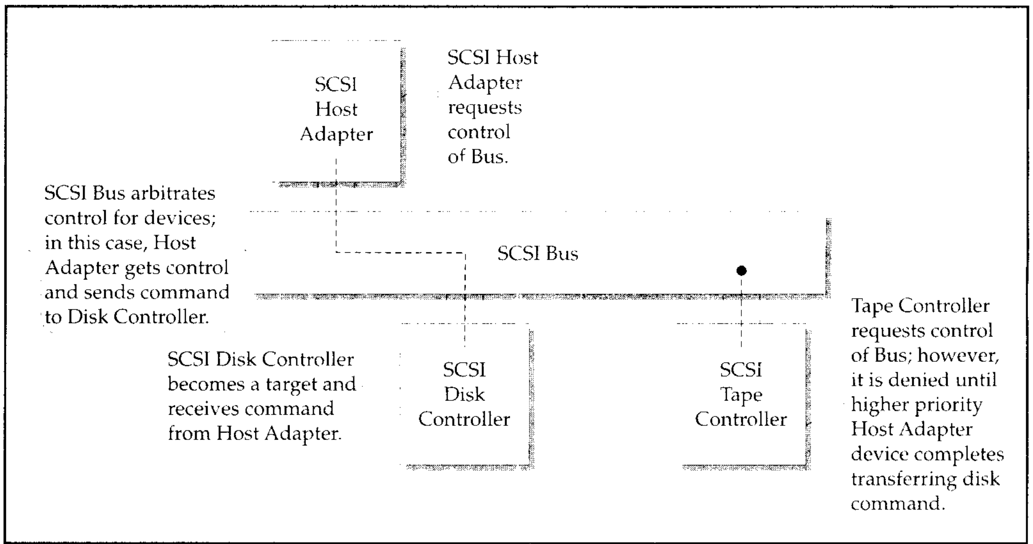
SCSI Host Adapter
SCSI Host Adapter requests control of Bus.
SCSI Bus arbitrates control for devices; in this case, Host Adapter gets control and sends command to Disk Controller.
SCSI Bus
SCSI Disk Controller becomes a target and receives command from Host Adapter.
SCSI Disk Controller
SCSI Tape Controller
Tape Controller requests control of Bus; however, it is denied until higher priority Host Adapter device completes transferring disk command.

Figure 7-3. *Bus operations: arbitration for control*

subjecting them to an increased overhead of ERC (error, recovery, and correction), length limitations, and sensitivity to external noise. Of the parallel connections, there are two types: the single ended bus and the differential bus. The single ended bus is just what it describes, a connection point where devices are chained together with a termination point at the end. On the other hand, a differential bus is configured more like a circuit where each signal has a corresponding signal. This allows for increased distances and decreasing sensitivity to external noise and interference among the multiple wires that make up the cable. Of these two parallel connections, differential will be the more effective for high-end storage systems; however, they are also more costly given the increases in function and reliability.

Serial connections are much simpler, consisting basically of two wires. Although they pass data in serial fashion, they can do this more efficiently with longer distances, and with increased reliability. High-speed network communications transmission media such as ATM and fiber optics already use serial connections. Wide buses that use parallel connections have the potential for cross-talk interference problems, resulting in intermittent signaling problems, which ultimately show up in reliability and performance issues for storage operation. The use of two wires allows substantially improved shielding. Although it may seem counter-intuitive, serial connections can provide greater bandwidth than wide parallel connections. The simpler characteristics of a serial connection's physical make-up allow a clock frequency to be increased 50 to 100 times, improving the throughput by a multiple of 200.

Differences in Bus and Network Architectures

Computer networks connect various devices of individual characteristics so they may communicate with one another. Some networks are peer-oriented, where all the devices are deemed equal and vie for network transport resources, transmission, and reception. Other networks (the majority of networks in use today) are hierarchical, where some devices have control and authority over other devices regarding who and how they can communicate, when they can transmit, and when and what they can receive. However, both architectures all share similar functions in that they transmit and receive data according to some type of predefined standard.

This demonstrates two distinct differences from bus architectures. First, networks have a defined transmit and receive function. This allows networks to continually pass data into the network path and receive data without regard for control and operation of the connection. This is different than a bus where strict flow control and arbitration takes place with devices assuming control over the path. Second, networks enable data transmissions over a long distance. Networks provide computers a method of transmitting and receiving data over great distances without regard to the underlying communications infrastructure. This requires network architectures to encapsulate the data to be sent over various types of media and transmission circuitry.

For the majority of networks today this is accomplished through a defined standard called TCP/IP. As we mentioned previously, this is a layered approach to communication functions that address all the requirements of transferring data from one device to another while making the communications transparent to the computer's application. Figure 7-4 shows the standard layers of TCP/IP. However, underlying the complexities of these functions is the simple concept of encapsulation and transmission. TCP provides the functions for taking the data from the application and OS and wrapping them in a form that will fit into the IP envelope. IP then acts as the envelope by putting the TCP wrapper into the IP packet (nee envelope) and sending it into the network with the appropriate addressing information.

The reverse happens on the receiving end. The IP functions receive packets addressed to its host device and pass these to TCP. TCP unwraps the data and presents it to the receiving application.

IP communications take place via multiple types of physical media. The media affects the integrity of the communications, the speed of the transmission, and the distances supported. Regardless of the media, the electronic standard that guides transmissions at the wire level is Ethernet. Working from packet encapsulation architecture, data packets once they hit the network vie for transmission time, resources, and priority. Obviously, the most prominent performance characteristic is the size of the packet. Packet size is determined by the implementation of Ethernet standards. These standards, as they are implemented in vendor network devices, such as NICs, switches, and routers, support this physical level of interface to the network. Capacities differ from standard Ethernet connections to the multi-gigabit capacity for Gigabit Ethernet (Gbe).

Most application transactions that access the network (this includes the application data, overhead of TCP/IP, and network error, recovery, and correction) will exceed the packet size, even for Gbe networks. So, moving data from a server to a client or from a server to a backup device can utilize larger and larger quantities of network bandwidth. Given this condition, nearly all remote application transactions will be accomplished through multiple packet transmissions.

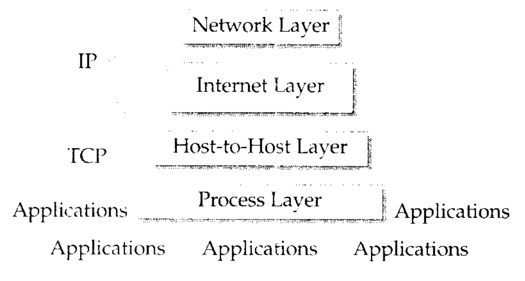

Figure 7-4. *The standard TCP/IP layers and functions*

Similar in fashion to the bus, its bandwidth, addressing, and interrupt functions also drive the network. However, major differences show up in interrupt functions (in other words, how do I get access to the network and utilize its resources?). Additional differences show up in the capacities of bandwidth, given the diversity of physical media that networks have to traverse. Physical addressing, an integral part of a network's topology, also becomes diverse, driven by the particular implementation of the networking standard a vendor's devices are supporting (for example, fast Ethernet and gigabit Ethernet).

The PCI Bus

As mentioned previously, today's standard for connecting peripherals into the motherboard is the Peripheral Component Interconnect (PCI). The operation is straightforward with the PCI bus. PCI determines, through the use of the PCI controller, the destination of the data. The destination could be local or to an expansion slot. If it is destined to an expansion slot address, the host adapter takes over and translates the protocol from PCI to host adapter protocol. This could be IDE, SCSI, USB, or Firewire.

The importance of the PCI bus is that it decouples control of the data path to the PCI bus itself. Therefore, it puts more performance responsibility on the PCI bus components, and relies on its bandwidth and speed characteristics. Most PCI buses today have a 32-bit bandwidth with various clock speeds ranging from 33MHz to 1GHz. The other importance of the PCI bus is its mezzanine architecture, whereby expansion of the bus itself can take place, extending the scalability of the number of peripherals connected.

The SCSI Bus

The most popular storage bus in use today is the SCSI bus. The Small Computer System Interface (SCSI) is a standard that allows the connection of various devices through a parallel interface. Although the standard calls for support of multiple devices, general usage has been to externally connect disk and tape systems to the server. Given the bandwidth, addressability, and speed, it has become the defacto standard for the external connection of storage devices into open computing systems encompassing both UNIX and Windows.

The SCSI bus allows connections from eight to sixteen devices in a single bus. The operation is through initiator and target architecture. Only two devices can use the bus at one time. The initiator gains control of the bus and transfers the command to a device on the bus. The receiving device is the target and processes the command. It then sends a response back to the initiator. SCSI devices connected to the bus are identified by their SCSI ID, which also serves as its address. The ID identifies the device's priority for arbitration—0 being lowest and 7 or 15 being the highest.

It is possible for one server to have multiple host SCSI adapters with access to multiple SCSI buses. Although these, by convention, will have different SCSI IDs, they are completely separate as they communicate through the host adapter to the PCI bus. As previously discussed, the software drivers assign the SCSI ID's. However, the SCSI controller provides the actual access to the physical device. The controller does this by assigning each device a Logical Unit Number (LUN). This, in effect, virtualizes the devices that are connected behind the controller.

Figure 7-5 depicts the most common SCSI configurations where servers with a single initiator support multiple targets (such as controllers). In this configuration, controllers support multiple LUNs or disks attached. Figure 7-6 also depicts a configuration with multiple servers supporting multiple targets. Although each target or controller is unique, multiple LUNs are addressed with similar LUN addresses. This works because the controllers own and direct each of their LUNs.

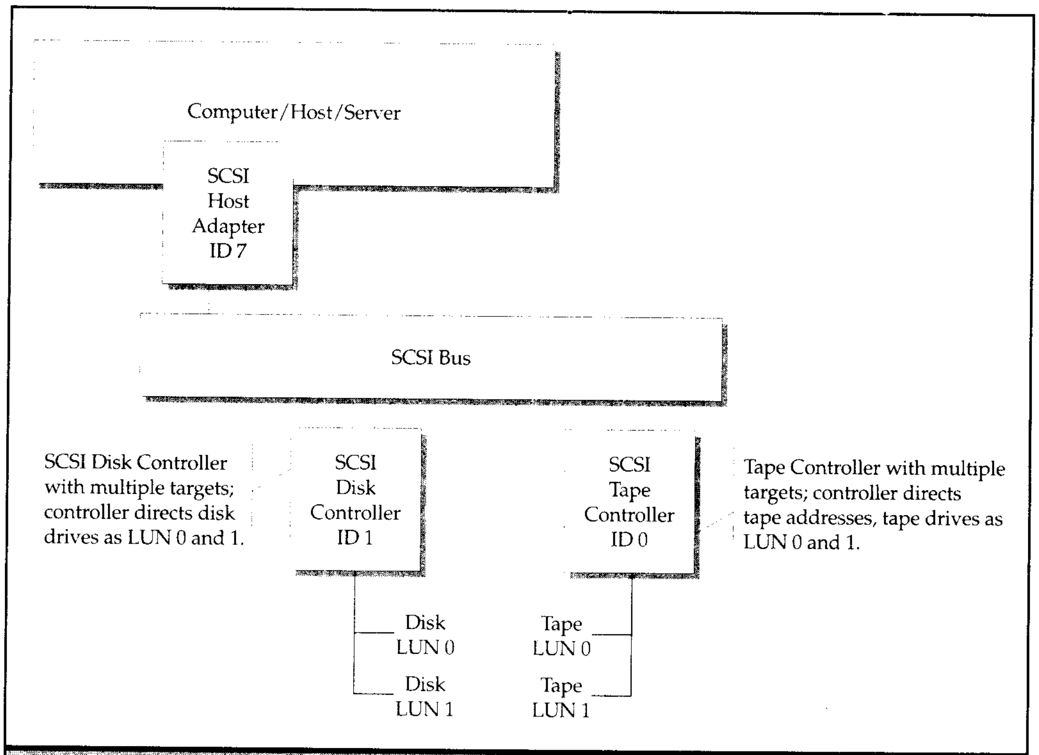
Computer/Host/Server
SCSI Host Adapter ID 7
SCSI Bus
SCSI Disk Controller with multiple targets; controller directs disk drives as LUN 0 and 1.
SCSI Disk Controller ID 1
SCSI Tape Controller ID 0
Tape Controller with multiple targets; controller directs tape addresses, tape drives as LUN 0 and 1.
Disk LUN 0
Disk LUN 1
Tape LUN 0
Tape LUN 1

Figure 7-5. *SCSI configuration single host*

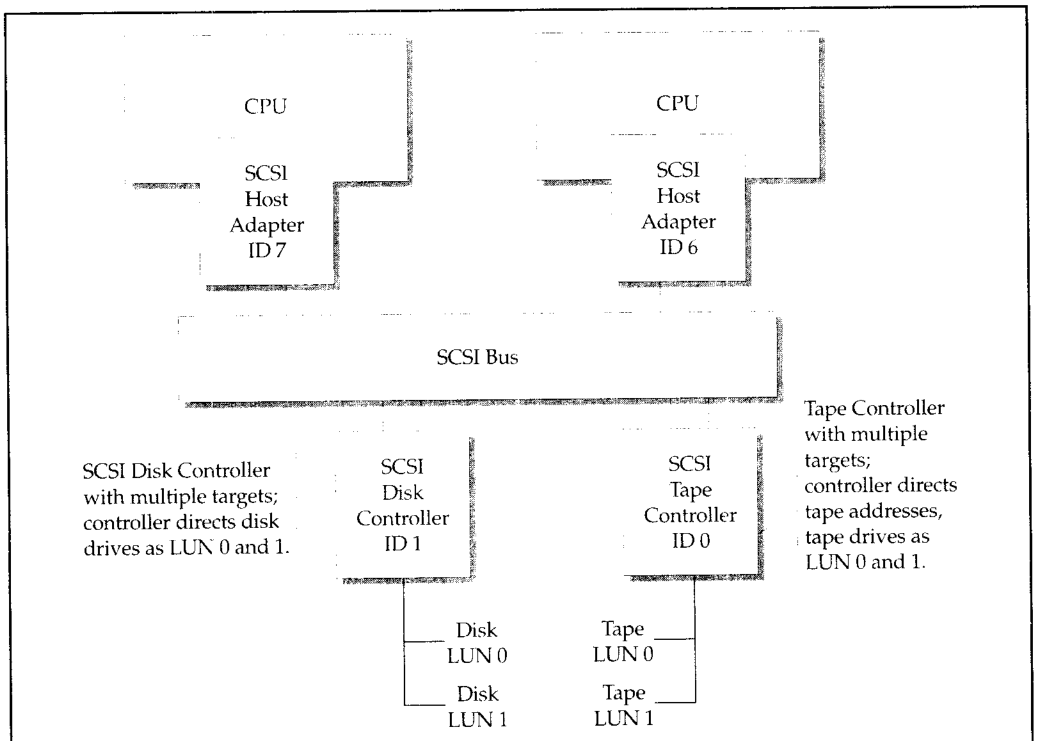
CPU
SCSI Host Adapter ID 7
CPU
SCSI Host Adapter ID 6
SCSI Bus
SCSI Disk Controller with multiple targets; controller directs disk drives as LUN 0 and 1.
SCSI Disk Controller ID 1
SCSI Tape Controller ID 0
Tape Controller with multiple targets; controller directs tape addresses, tape drives as LUN 0 and 1.
Disk LUN 0
Disk LUN 1
Tape LUN 0
Tape LUN 1

Figure 7-6. *SCSI configuration multiple hosts*

Fibre Channel

It's a network... it's a bus... no, it's Fibre Channel. Fibre Channel is a layered connectivity standard that has the characteristics of a bus, the flexibility of a network, and the scalability potential of MIMD (multi-instruction multidata) configurations. In implementation, it uses a serial connectivity scheme that allows for the highest level bandwidth of any connectivity solution available today. This architecture allows for implementations to reach a burst rate as high as 200 MBps for I/O operations, with aggregate rates depending on workload and network latency considerations. Regardless of the latency issues, this is a tremendous enhancement to throughput over traditional bus connectivity.

Note *Multi-instruction multidata computer taxonomy (known as MIMD) occurs where multiple instructions are executed with operations on multiple data sources. This has become synonymous with MPP computer configurations where parallel processing at the CPU level and I/O levels exemplify this theoretical term.*

To say that Fibre Channel is a network architecture or that Fibre Channel is a serial bus connectivity standard is an exercise for the esoteric debating society. Discussion here will result in nothing more than a last word at the science club meeting. In all fairness to the innovations surrounding Fibre Channel (FC), it was designed incorporating the characteristics of both to meet the growing scalability requirements of computer connections. It has accomplished this task in particular with storage systems.

The FC standard is implemented using a layered protocol consisting of five levels. As depicted in Figure 7-7, the first three levels deal with the physical attributes. As shown here, these are FC-0 through FC-2. These layers control the lowest level operations, from media types supported, to signal encoding for transmission, to the encapsulation and transmission control functions. FC-3 through FC-4 provide attributes for common services and mapping to other protocols. The most popular so far being the SCSI and IP mappings that support storage system operation and remote linking to IP networks.

FC can be implemented through three different topologies, as indicated in Figures 7-8, 7-9, and 7-10. These range from the most simple, point-to-point configurations, to the more complex, albeit more powerful, fabric implementation. The Fibre Channel Arbitrated Loop

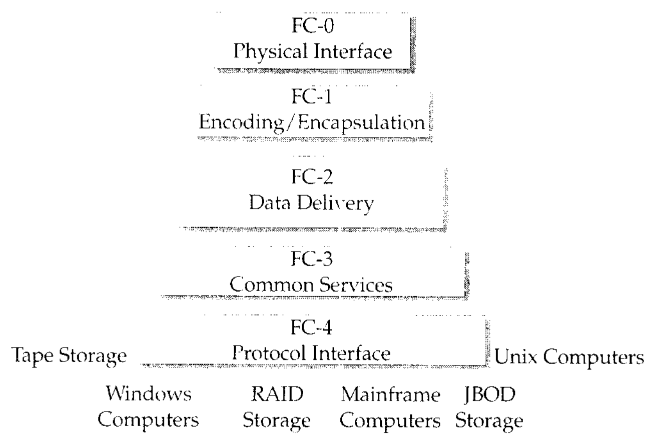

Figure 7-7. *Fibre Channel layered architecture*

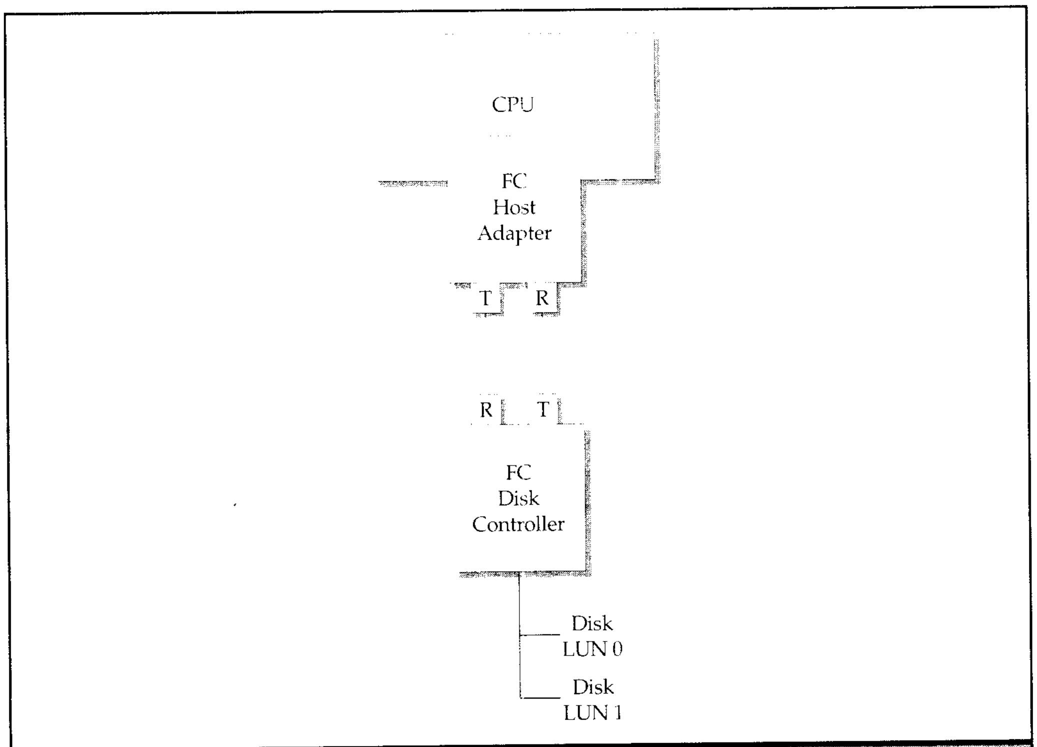

Figure 7-8. *The FC point-to-point topology*

(FC-AL) topologies have served as a transitional move as FC storage systems moved from FC Hub configurations, where the FC-AL provided access to the bandwidth benefits of the protocol; however, the loop arbitration overhead sacrificed most of these enhancements.

Fibre Channel Operation

FC operates on a serial link design and uses a packet type of approach for encapsulation of the user data. FC transmits and receives these data packets through the node participants within the fabric. Figure 7-11 shows a simplified view of the FC fabric transmission from server node to storage node. The packets shipped by FC are called frames and are made up of header information that include addresses, user data (encapsulating an incredible 2,192 bytes per frame), and error recovery code (ERC) information.

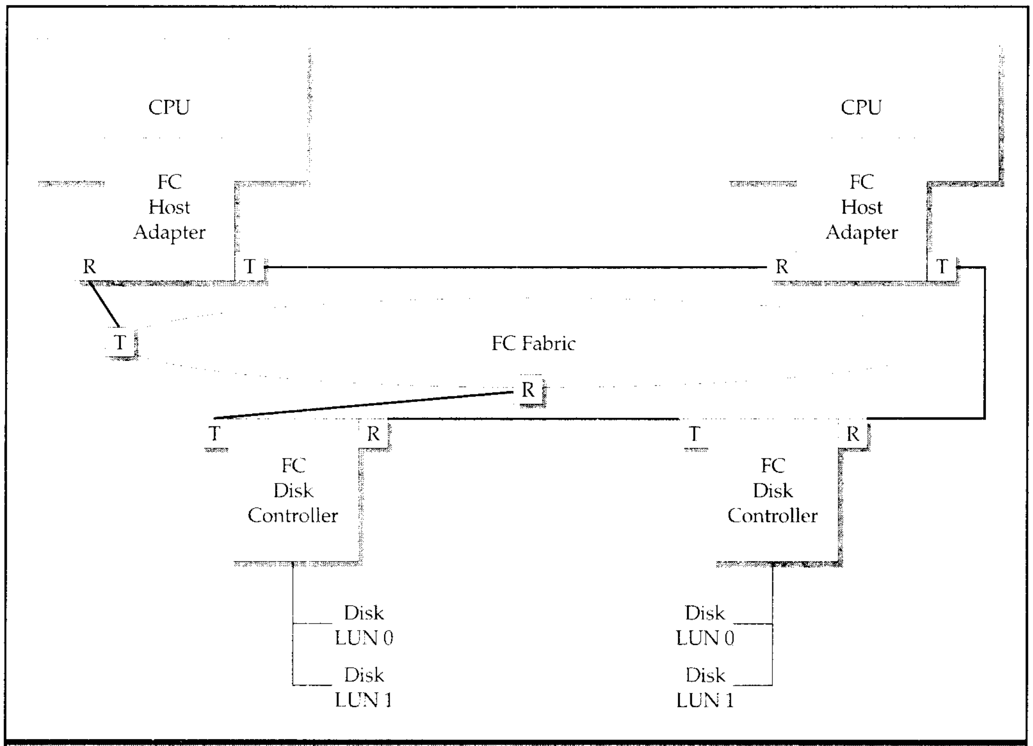

Figure 7-9. *The FC arbitrated loop topology*

Each device that uses the FC Fabric is a node. FC provides both flow control and addressing for nodes when they contact and log in to the fabric, as illustrated in Figure 7-11 where Node 01 is assigned the port address of N01-fabric1. This is the N-Port address for the external device, in this case a Host Adapter for a server. This is connected to Fabric port F-01 of the switch's fabric (in other words, the fabric switch with an address of F01-fabric1 is shown as a fabric port). The storage system is connected using the same pattern.

The devices are controlled using calculations of available buffer space within each device as well as knowledge of time sequences, thereby keeping frame overflows to a minimum. Finally, the fabric itself executes under a predefined and supported class of service. Of these classes, only two are appropriate for typical storage systems: Class 2 and Class 3. These define a network connectionless service this way: acknowledgement of transmission is designated as Class 2; no notification of transmission is displayed as Class 3.

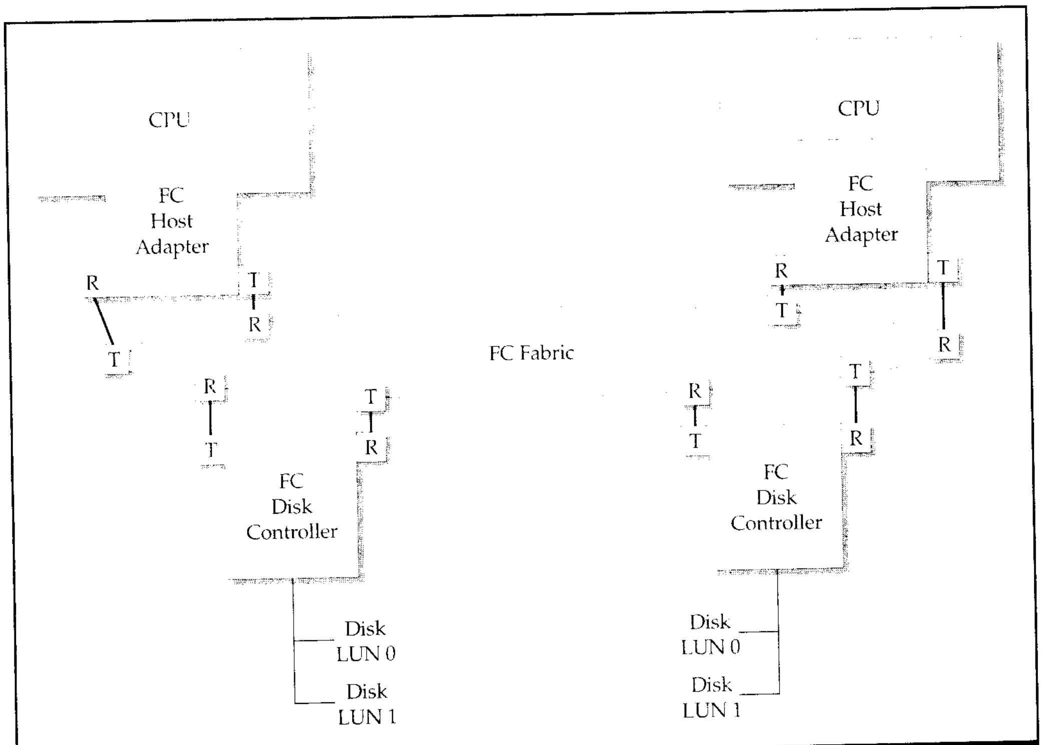

Figure 7-10. *The FC fabric topology*

Other I/O Buses

As peripheral devices increased in both number and diversity, the need to connect these to the computer system quickly and efficiently also grew. With devices such as digital cameras, larger external and removable storage devices, scanners, printers, and network options, the diversity and flexibility of the traditional computer bus was expanded to include support for all these devices. The systems and storage industry responded with innovative solutions that enhanced and implemented the traditional computer bus with serial technologies and higher speed protocols such as Firewire. In this section, we discuss two of the most popular bus evolutions: the Universal Serial Bus (USB) and Firewire. USB and Firewire have been successful solutions in the workstation and PC industry. Other evolutions to the traditional bus architecture are evident in the opposite side of the spectrum in the super computer and parallel processing industries. These technologies, driven by complex computational problems, have resulted in creative architecture that bind multiple computers together applying their combined power. These architectures are characterized by their creative bus implementation in

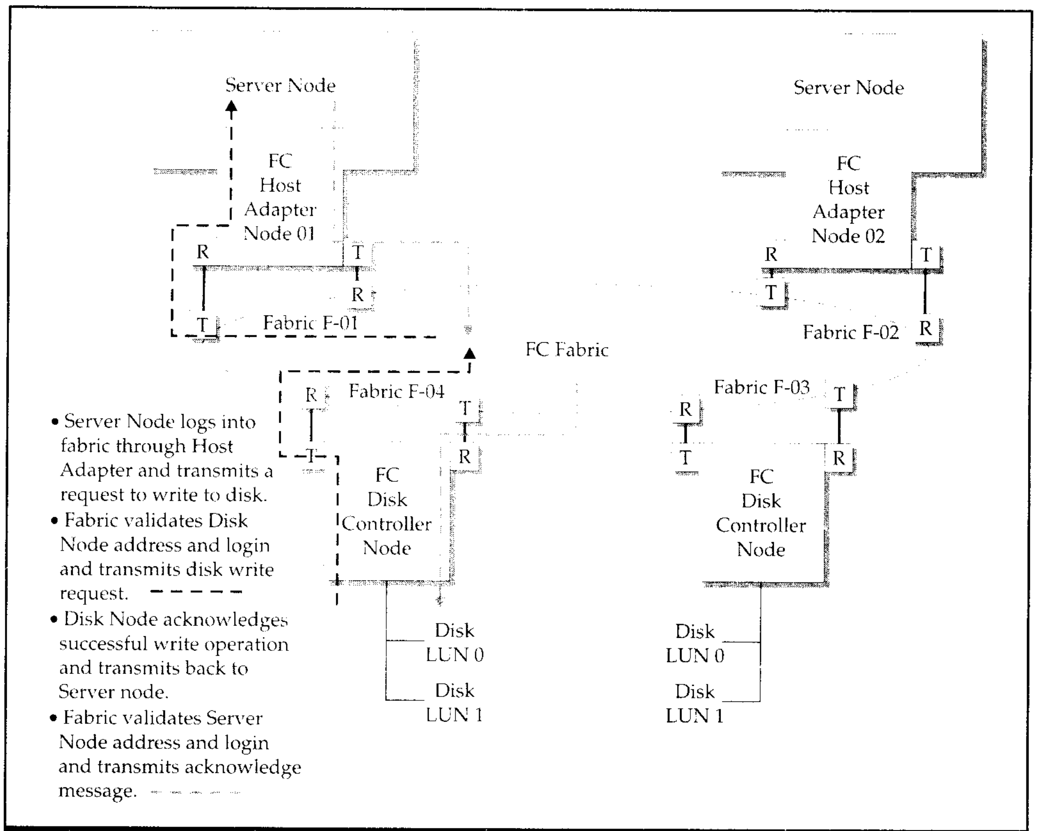
Server Node
FC Host Adapter Node 01
R
T
R
T
Fabric F-01
FC Fabric
Server Node
FC Host Adapter Node 02
R
T
T
Fabric F-02
R
Fabric F-04
R
T
T
R
FC Disk Controller Node
Disk LUN 0
Disk LUN 1
Fabric F-03
R
T
T
R
FC Disk Controller Node
Disk LUN 0
Disk LUN 1
• Server Node logs into fabric through Host Adapter and transmits a request to write to disk.
• Fabric validates Disk Node address and login and transmits disk write request. - - - - -
• Disk Node acknowledges successful write operation and transmits back to Server node.
• Fabric validates Server Node address and login and transmits acknowledge message. - - - - -

Figure 7-11. *FC operations*

connecting hundreds of processors and peripheral devices together. We summarize some general configurations in our discussion of these creative bus strategies.

USB and Firewire

Like the SCSI standard, the Universal Serial Bus (USB) and IEEE1394 standard, Firewire, was developed into a product to serve the multiple and diverse types of external devices that can interface with a single PC or workstation user. Used almost entirely for single user systems, the bus concept is similar to SCSI in terms of a host adapter translating PCI communications to USB or Firewire adapters whose protocols communicate within serial bus architectures. One of the differences is the support for the diverse and often disparate number of peripheral devices requiring connection to single user systems. These devices range from mice to joysticks to optical scanning equipment. With the exception of CDs, storage has come late to this area and remains lacking in providing levels of functionality needed for server level implementation.

Both of these solutions are ways of expanding the I/O capability of computers; however, the use of USB or Firewire disks for large-scale commercial use has not taken hold. Although this came about for many reasons, it is primarily due to the direction the USB standard took to support asynchronous and isochronous data transfers within a half duplex architecture. This limited performance for any bandwidth gains provided by a serial connection. In addition, disk manufacturers have not supported the serial interface and command structure used with disk drive functions. Finally, the limiting number of devices that can be addressed and the inability to deal with advanced storage features such as controller/LUN functions and RAID partitioning will continue to orient USB toward the single-user PC market.

Creative Connection Strategies

There are many ways of connecting peripherals to the server. Using the bus, network, and hybrid technologies that we discussed earlier, these strategies/implementations have evolved from the exotic innovations of supercomputing, alternatives to symmetrical multiprocessing (SMP), and developments within distributed computing architectures. All of these provide some type of high-speed interconnect between computing nodes that enable increased parallel computing tasks, allow support for larger workloads and increased data sizes, and which give way to increased overall performance scalability. These interconnects can be characterized as implementations of various bus and network technologies that produce non-standard configurations. Generally, these implementations demonstrate characteristics of a network with sophisticated data distribution functions that leverage specific CPU architectures (for example, IBM RISC-based systems, SUN Sparc systems, and Intel CISC systems).

There are three general types of high-speed interconnect topologies that have been commercialized. These are the shared nothing, shared I/O, and shared memory models. These models supplement the basic components of computer systems (for instance, RAM, CPU, and internal Bus), and provide an extension to a tightly coupled or loosely coupled set of distributed computer systems.

Shared Nothing

Figure 7-12 illustrates the configuration of a shared nothing system. These systems form the foundation for Massively Parallel Processing (MPP) systems, where each computer node is connected to a high-speed interconnect and communicates with nodes within the system to work together (or in parallel) on a workload. These systems generally have nodes that specialize in particular functions, such as database query parsing and preprocessing for input services. Other nodes share the search for data within a database by distributing it among nodes that specialize in data access and ownership. The sophistication of these machines sometimes outweighs their effectiveness, given that it requires a multi-image operating system (for instance,

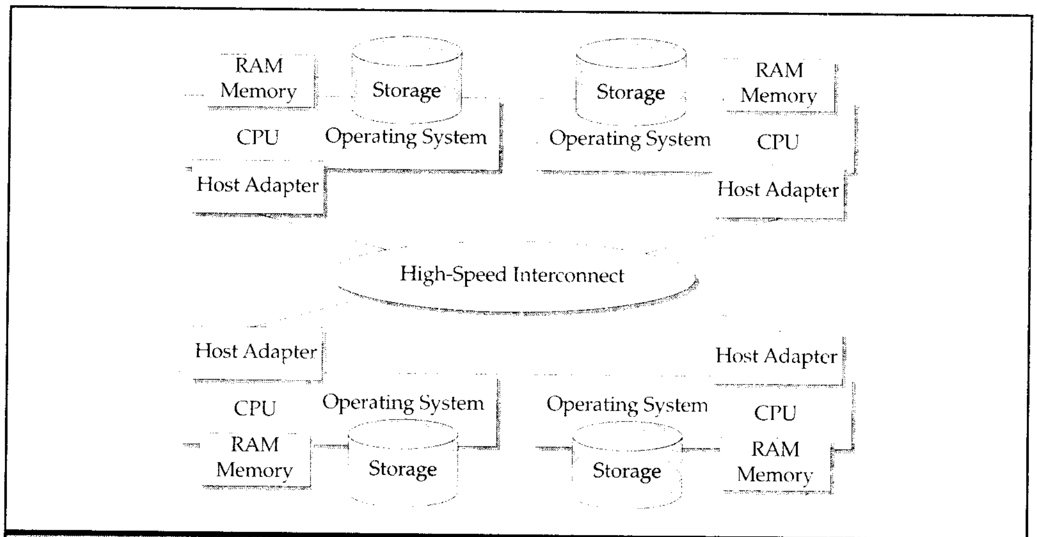

Figure 7-12. *A shared nothing high-speed interconnect*

an OS on each node), sophisticated database and storage functions to partition the data throughout the configuration, and the speed, latency, and throughput of the interconnect. In these systems, both workload input processing and data acquisition can be performed in parallel, providing significant throughput increases.

Shared I/O

Figure 7-13 indicates the configuration of a system that has shared I/O. Although more common than the MPP machines, the shared I/O requires the computer systems to share a common I/O bus and therefore extend the capability to provide additional access to larger amounts of data. Operation of these machines requires that a single image operating system control the operation of I/O and application processing across the computing nodes and shared I/O. These systems offer enhanced access to large amounts of data where I/O content can be large and multiple computing nodes must operate in parallel to process the workload.

Shared Memory

Figure 7-14 shows the most common of alternative interconnects: the shared memory model. This makes up most of the high-end SMP machines where multiple CPUs share a large RAM, allowing CPUs to process workloads in parallel with enhanced performance from a common RAM address space that's both physical and virtual. Also operating under a single image operating system, these systems enhance the capability to process workloads that are high-end OLTP with small I/O content.

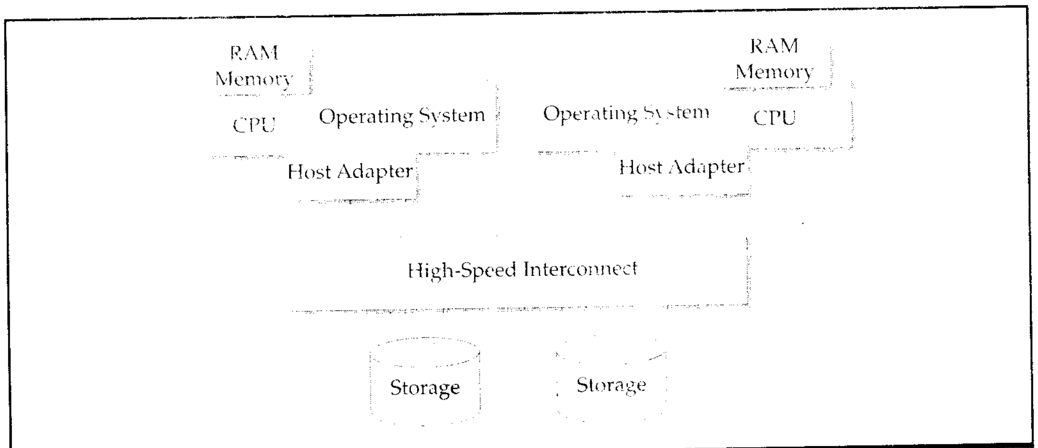

Figure 7-13. *A shared I/O high-speed interconnect*

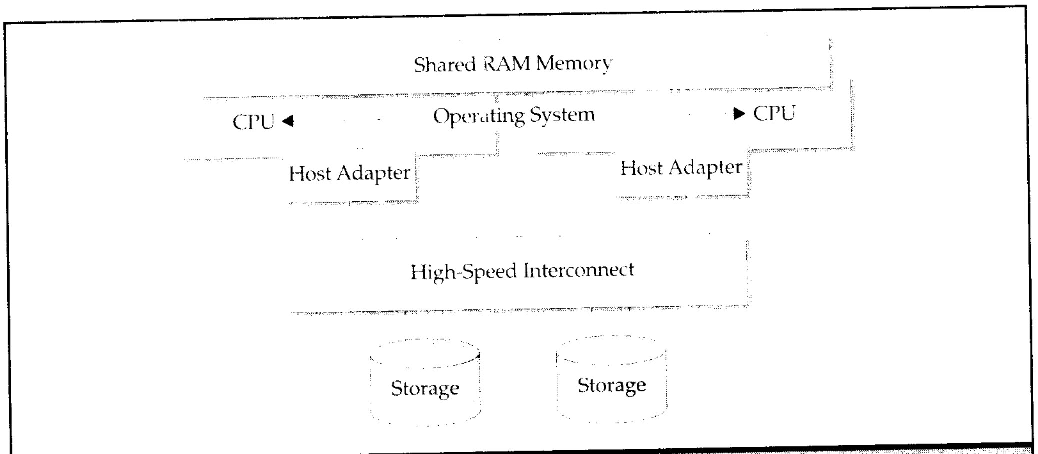

Figure 7-14. *A shared memory high-speed interconnect*