Chapter 20

Input/Output Subsystem

The CM-5 system achieves true I/O scalability by connecting I/O devices directly to the CM-5 Data Network, achieving three important goals of the I/O system:

- Any I/O device, or collection of I/O devices, has the bandwidth needed to achieve high throughput. If a device needs more bandwidth than one Data Network connection can supply, it is given multiple connections.
- A system's I/O storage capacity is independent of the system's computational capacity: I/O capacity is expanded merely by adding Data Network connections. Since the bandwidth of the CM-5 Data Network grows linearly with the number of connections, the performance of the Data Network expands to meet the needs of the additional I/O devices.
- I/O resources are shared equally. Any partition or any computer connected to the CM-5 by a LAN may access all CM-5 I/O devices. I/O devices may also communicate directly with one another, facilitating, for example, direct disk-to-tape copies without the use of partition resources. In addition, the Data Network allows multiple partitions to perform I/O operations simultaneously, without affecting each other's performance or data integrity.

In addition to scalable bandwidth, the CM-5 I/O system supports industry software standards, such as UNIX file system access and TCP/IP networking with other machines. The CM-5 operating system (CMOST) provides straightforward, consistent methods for performing I/O and accommodates a wide range of I/O devices, such as the Scalable Disk Array, Integrated Tape System, HIPPI and FDDI network connections, and Thinking Machines' CMIO bus devices. These CMIO bus devices, such as the DataVault mass storage system, are supported both on the CM-5 and on the CM-2 and CM-200. They thus allow multiple Connection Machine systems to share I/O devices and to have direct access to the data in those devices.

To a user application, the CM-5 I/O system consists of a collection of virtual I/O devices, any of which can be home to a file system. File formatting and physical characteristics of the devices are invisible to the application code. Implementation details for each I/O device are hidden by a combination of hardware and software interface modules.

20.1 I/O Architecture

Every I/O device is connected to the CM-5 through the Data Network, with each I/O interface occupying a block of Data Network address space. By convention, all I/O devices occupy the upper region of that address space. (See Figure 41.) An I/O interface attaches to the Data Network through one or more Network Interfaces — the same type of interface that connects processing nodes to the Data Network.

An important consideration in any I/O scheme is matching the system's internal bandwidth to the I/O rates of peripheral devices attached to the system. Here, the Data Network's intrinsic scalability plays a critical role. The number of Network Interfaces used to attach an I/O device to the Data Network determines how much of the network's bandwidth is made available to the device. The more ports an interface has into the Data Network, the greater its potential bandwidth.

An I/O interface with a single Network Interface provides a nominal bandwidth of 20 Mbytes/sec across the Data Network. This capacity can easily accommodate low- and medium-speed I/O devices. Interfaces for high-performance peripherals are implemented with as many Network Interfaces as are needed to support the transfer rates required by the particular device. The Scalable Disk Array, for example, has one Network Interface for each Disk Storage Node; when a Storage Node is added for increased capacity, a Network Interface is added also, increasing the device's aggregate transfer rate. The CM5-HIPPI, which can provide 160 Mbytes aggregate bandwidth, has eight Network Interfaces.

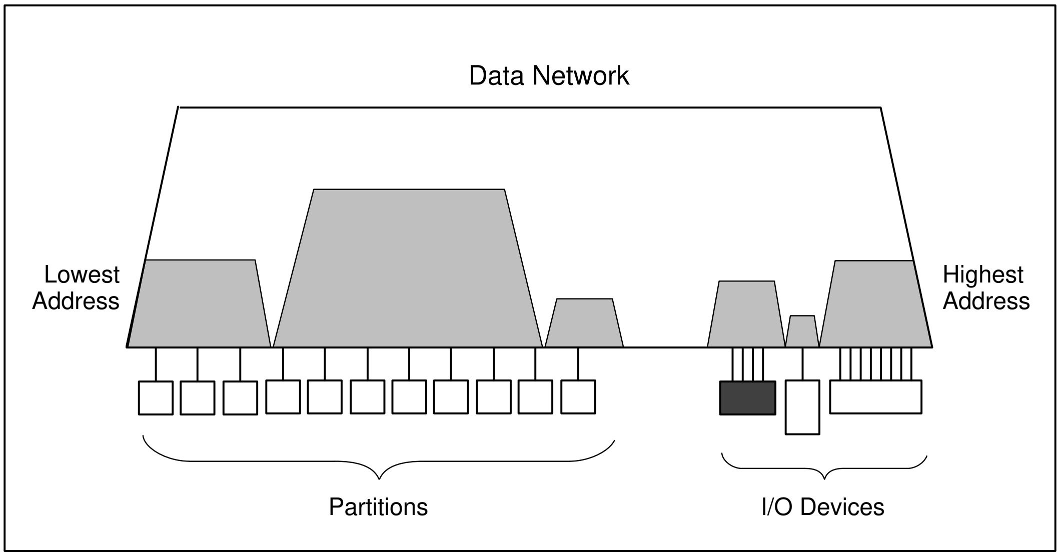

Figure 41. CM-5 I/O subsystem block diagram.

Each I/O interface requires a control processor to act as its file server and supervise all I/O operations for that device. For the Scalable Disk Array, the processor is known as an I/O control processor (IOCP). The IOCP has the same capabilities as other CM-5 control processors, and so may be used for other functions, as well as for file server functions. For the Integrated Tape System, the Connection Machine HIPPI devices, and the CMIO bus devices, the file server is embedded within the device itself.

20.2 File System Environment

The CM-5 system can access three file systems: SFS, CMFS, and UNIX. SFS (Scalable File System) and CMFS (Connection Machine File System), both proprietary to Thinking Machines Corporation, exploit the great speed and massive storage capabilities of the Connection Machine I/O systems. The UNIX file system further enhances system usability. All three file systems organize files into directories, use pathnames to identify them, and treat all I/O devices as files. Following is a brief description of each file system:

- SFS is NFS-mounted on an IOCP and manages the files stored on a Scalable Disk Array or Integrated Tape System. The SFS file system is a fully compatible enhancement of the UNIX file system, with extensions to support parallel I/O and much larger files than most UNIX implementations can accommodate. A CM-5 program can access the SFS file system via CM Fortran, C*, the CMMD library, a subset of the CMFS library and the CMFS commands, and standard UNIX routines and commands.

  From a user's perspective, the SFS file system behaves like a UNIX file system: for example, it stores each file's data in canonical serial order.

- CMFS is a UNIX-like file system that can reside on the CMIO-bus data-storage devices. Like the SFS file system, the CMFS file system has extensions to support parallel I/O and very large files. A CM-5 program can access the CMFS file system via the same interfaces that are used for the SFS file system (to use the UNIX commands and calls, however, the CMFS file system must be NFS-mounted).

  From a user's perspective, although the CMFS file system is similar to the UNIX file system, there are some differences in its appearance and behavior. For example, there are special environment variables that apply only to the CMFS file system.

- The standard implementation of the UNIX file system can reside on all CM-5 control processors and on all other serial computers in Connection Machine systems.

Each file system is completely separate from all others, having a separate directory tree and its own current working directory.

Regardless of which file system is being accessed, all I/O transactions are modeled as reads and writes to files. UNIX-style **`open()`** and **`close()`** requests go to the file server and thus are independent of the file system data storage implementation. Requests for other file system operations, such as reading and writing files, also go to the file server, which then directs the transfer of data. In these cases, data may be transferred in parallel directly from the source to the destination, without passing through the file server. For the SFS file system, data and control information travel separately through the Data Network. (See Figure 42.) For the CMFS file system, data travels through the Data Network, while control information travels over the Ethernet. For the UNIX file system, data and control information both travel over the Ethernet.

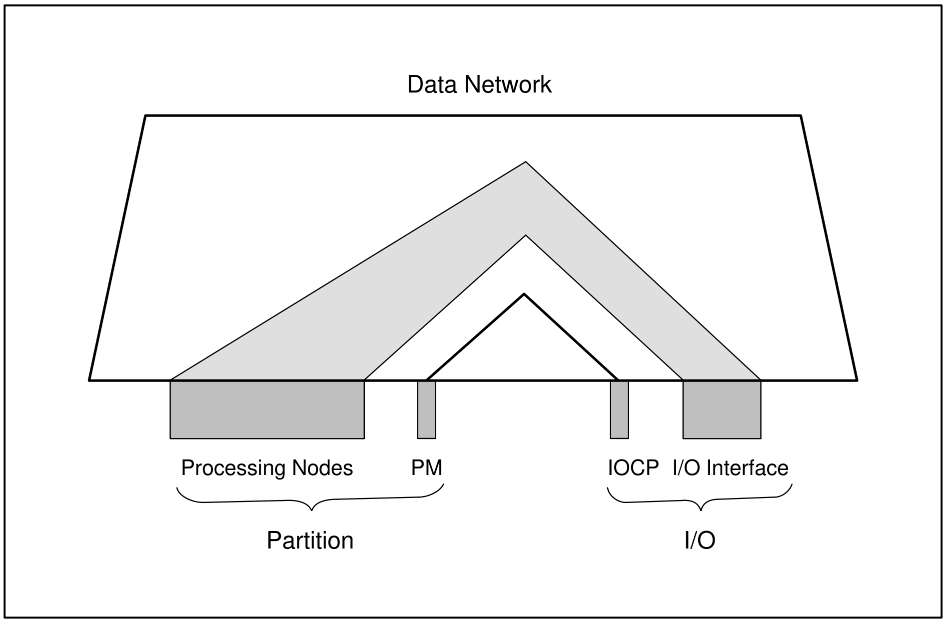

Figure 42. Independent control and data paths through the Data Network.

When an application program requires I/O services, the partition on which the application is running initiates the file transfer with an appropriate read or write command. It directs the I/O request to the appropriate file server, which assumes control of the transfer.

For a **`read()`** operation, the file is retrieved from the I/O device, encapsulated in message packets, and sent through the Data Network to the partition that requested the data. File order information embedded in the message packets enables the receiving partition to arrange the file data in correct sequence within each processor. A **`write()`** operation is similar but reverses the flow of data.

Different versions of **`read()`** and **`write()`** are used, depending on whether an application is running on a single processor or on a set of parallel processing nodes. A serial application uses the conventional UNIX **`read()`** and **`write()`** commands. Parallel applications use parallel read and parallel write routines.

Multiple I/O devices may be logically ganged for striped operation as a single file system. The SFS file system automatically routes data between requesting processors and individual I/O devices so that all striping is transparent. This same facility makes SFS file structure independent of the number of computational processors that read or write the data and transparent to programmers.

20.3 CM-5 I/O Interfaces and Device Implementation

This section describes the SDA, ITS, and CM5-HIPPI, the key elements of the CM-5 I/O subsystem implementation. The CMIO bus peripherals and the two standard bus interfaces (SVME and SBA) are described in Section 20.4.

20.3.1 Scalable Disk Array

The Scalable Disk Array (SDA) is an extremely high performance, highly expandable RAID-3 disk storage system composed of Disk Storage Nodes packaged within CM-5 cabinetry. The basic Disk Storage Node — providing 9.2 Gbytes of storage, a peak bandwidth over 17 Mbytes/sec, and 25 Mips of processing power — comprises a controller built on a SPARC processor, a Network Interface, a large disk buffer, four advanced SCSI controllers, and eight 3.5″ hard disk drives. (See Figure 43.) The drives are mounted in removable modules that facilitate installation and removal during servicing operations. These parts are widely available and represent mature high-volume technologies, thereby contributing to the high reliability of the SDA system. Additional custom hardware is provided to augment the transfer of data directly between the disks, the buffer, and the Data Network.

The SDA Disk Storage Nodes — analogous to the computational processing nodes offered with the CM-5 architecture — are directly connected to the CM-5's Data Network. The direct connection to the Data Network enables each Disk Storage Node to contribute not only to storage capacity but also to I/O performance; the number of the Disk Storage Nodes in an SDA system can be increased or decreased, thereby achieving an I/O system matched to the performance and capacity needs of the CM-5 applications.

For example, a single Disk Array Module comprising 3 Disk Storage Nodes provides 25 Gbytes of storage at I/O bandwidths of up to 33 Mbytes/sec sustained. Adding one more Disk Storage Node increases capacity to over 33 Gbytes and I/O bandwidth to 44 Mbytes/sec sustained.

The number of Disk Array Modules that can be installed in a CM-5 system is limited only by the number of address spaces the system contains. A full cabinet's worth of SDA — eight Disk Array Modules — uses 256 Data Network addresses and provides 200 Gbytes of storage capacity.

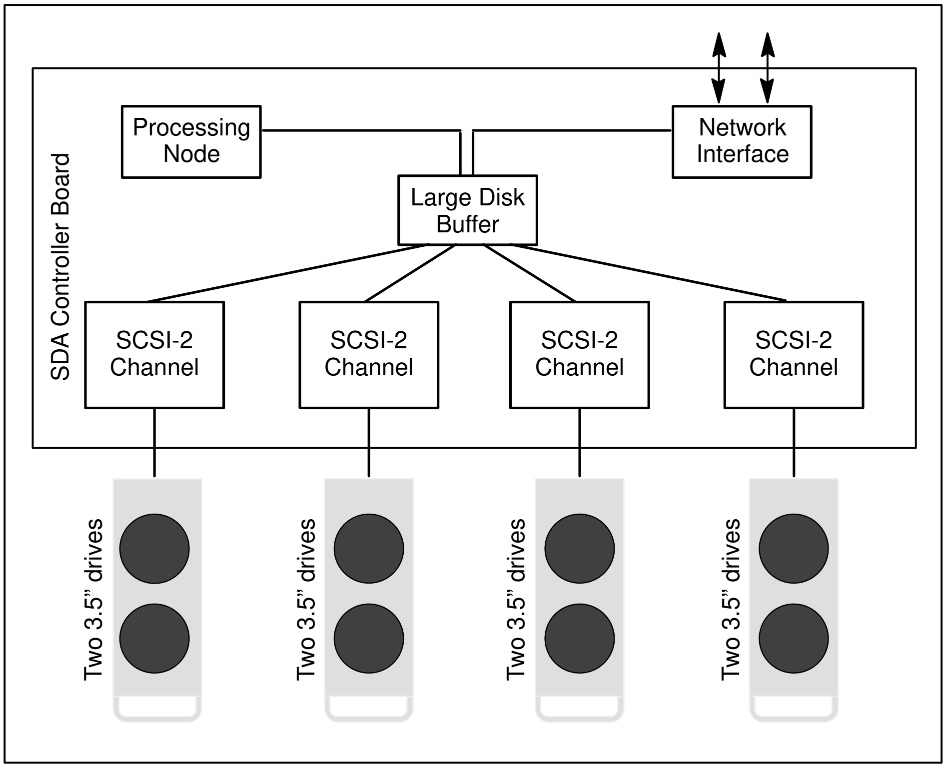

Figure 43. Block diagram of a Disk Storage Node.

Data Storage

Data stored on the Disk Storage Nodes is always available in serial order. The CM-5 stripes the data across all the Disk Storage Nodes in the system, so that each node contributes to the overall transfer rate available to the user. All disks in the SDA act together, in conjunction with the operating system, to transfer data simultaneously. In fulfilling a read request, for example, the operating system knows how an array is spread across the processors, and automatically distributes the data coming from the SDA to the correct locations. From the application's perspective, the SDA appears to be a single, high-capacity, high-performance disk.

Special-purpose hardware on the Disk Storage Node controller assists the operating system in handling data-ordering issues, providing a seamless mechanism for moving data between the Disk Storage Nodes, partitions of processing nodes, other CM-5 I/O devices, other serial computers, and I/O devices connected to them. This combination of hardware and software support relieves the applications programmer from the burden of dealing with data ordering, without compromising performance.

Availability and Serviceability

In addition to 22 data-storage disks, a Disk Array Module contains 1 parity disk and 1 spare disk. The parity disk stores redundant information — used to recreate the data should a disk fail — automatically generated by the operating system as a simple parity summing operation:

- On disk writes, parity is generated in the processing nodes and transferred with the data to the parity disk.
- On disk reads, all the data, including that from the parity disk, is sent to the processing nodes, which calculate and check the parity.

In the event of a drive failure, a sparing and healing operation is performed. *Sparing* is a software procedure that logically replaces the failed drive with the SDA's spare drive (provided for that purpose); *healing* is the process that reconstructs the corrupted data using the parity information and stores it on the spare drive. The entire operation takes about one hour, each phase of which is signaled by the LEDs.

The sparing and healing utilities, as well as utilities that facilitate other administrative tasks such as running SDA diagnostics and controlling the error-logging and -reporting mechanism, are provided through the SDA Command Center (SDACC). The SDACC is a menu-driven program that runs on the SDA's IOCP, allowing the system administrator to maintain the SDA in a directed manner with minimal down-time and no guess-work.

Failure Detection

Essential to maintaining high availability is the detection of failures in the data paths. Failures are detected by parity checking circuits located in all the major data paths and in all the controller and processor buffers. In addition, internal to the disk drives, extensive use of error-correcting codes provide for complete correction of bit-error bursts up to 48 bits in length, and detection of error bursts as long as 120 bits.

20.3.2 Integrated Tape System

The Integrated Tape System (ITS) is an industry standard–compliant tape-controller system that, like the SDA, is integrated into the CM-5 cabinetry, with direct connections to the Data Network. The ITS offers the same unlimited scalability as the other parts of the CM-5 system: dozens or even hundreds of ITS modules can be connected to a single CM-5, each accommodating, for example, two IBM 3480–compatible tape drives or eight 8mm drives. The drives can be located as far as 20 meters away from the CM-5 cabinet.

The ITS tape controller, shown in Figure 44, is built on a SPARC processor, a Network Interface, an 8-Mbyte data buffer, and two advanced 16-bit SCSI-2 channels. The ITS supports industry-standard tape formats such as 3480 square tapes (200 or 400 Mbytes per tape) and 8-mm cartridges (5 Gbytes per tape). The ITS tape controller can sustain a maximum throughput of 15 Mbytes/sec and can burst at up to 20 Mbytes/sec.

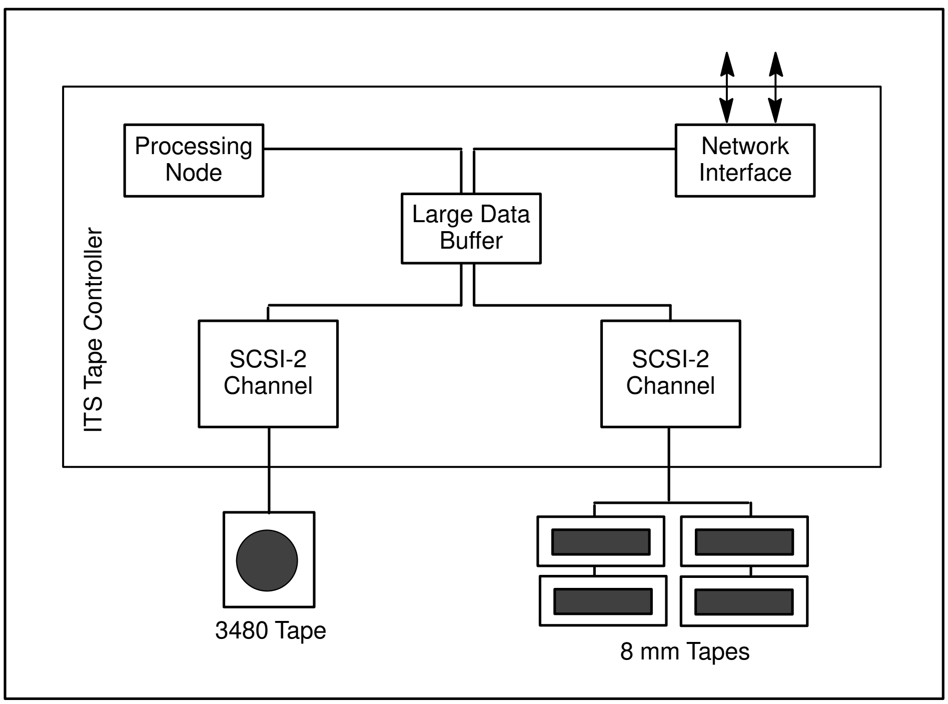

Figure 44. An ITS system. The tape devices can be located up to 20 meters away from the CM-5.

The ITS tape controllers are backplane-compatible with the SDA Disk Storage Nodes: ITS tape controllers and Disk Storage Nodes can be intermixed. The total number of each type of board installed in a system is limited only by available address space.

All software needed to control the ITS tape drives and negotiate the data transfers is included in a CMTape software package that includes standard UNIX tape facilities as well as CM system–specific functionality. Users gain access to the ITS either through CMTape commands executed from a UNIX shell or through CMTape utilities callable from applications. Included in the CMTape package are functions for performing:

- disk back-ups and restores
- label processing of IBM- and ANSI-standard label tapes
- common format conversions (for example, between EBCDIC and ASCII)

All these functions support file sizes greater than the 2-Gbyte restriction imposed by UNIX. In addition, applications for the Connection Machine models CM-2 and CM-200 CM-IOPG (CM I/O processor) can be run on the ITS with only minimal changes.

The SDA is used to stage data as it is written to the ITS from the CM-5 or read from the ITS to the CM-5. Transfers between disk and tape occur directly — no CM-5 partition resources are involved in the transfer.

20.3.3 Connection Machine HIPPI Interfaces

CM-HIPPI and CM5-HIPPI are bus interface controllers that are designed to transfer data at a high speed according to the ANSI HIPPI draft standard. The interfaces are primarily intended to link the CM-5 and its storage devices to other supercomputer systems. See Figure 45.

CM-HIPPI is a complete, integrated system that receives CMFS file system commands from a CM control processor over an Ethernet cable. The CM-HIPPI contains a CPU, disk drives, a VMEbus, and HIPPI input and output interface modules. The modules provide a full-duplex I/O interface between a pair of external HIPPI buses and a pair of internal buses, one each for incoming and outgoing data. The internal buses connect the input and output interface modules to up to eight HIPPI-to-CMIO interface modules via a set of multiplexing switches. Using an I/O bus adaptor backplane in the CM-5, the switches establish and break links between a CMIO bus and the HIPPI input and output ports. The peak aggregate I/O bandwidth of this configuration is 25 Mbytes/sec.

CM-HIPPI supports

- HIPPI-PH and HIPPI-FP
- Multiplexed access to a HIPPI connection

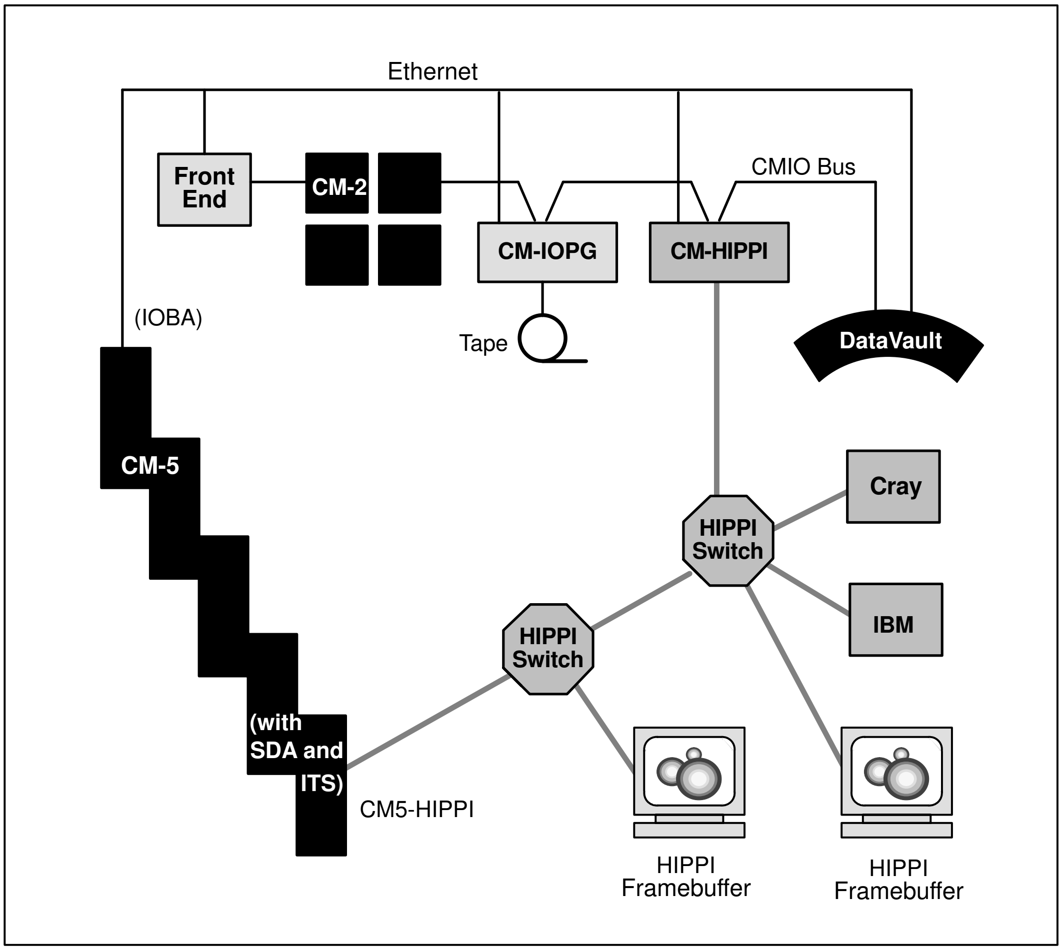

Figure 45. Typical HIPPI network with Connection Machine systems.

CM5-HIPPI occupies 1/8 of a standard CM-5 cabinet and provides a 32-bit source and destination interface to the I/O bulkhead of the machine. The CM5-HIPPI connects directly to the CM-5 Data Network to provide 160 Mbytes/sec aggregate bandwidth. There is an on-board diagnostic loopback between source and destination. In addition to the CM-HIPPI features, CM5-HIPPI supports

- Users' HIPPI framing protocols
- TCP/IP

20.4 CMIO Bus Device Implementation

For application portability, the family of CM-2 and CM-200 peripherals continues to be supported on the CM-5 system. These devices reside on Thinking Machines' proprietary CMIO bus, which is connected to the Data Network via the IOBA (I/O bus adapter). The CMIO bus peripherals are:

- **DataVault.** This is a high-performance, disk-based mass storage system. It allows applications running on a CM-5 to access as much as 60 gigabytes of random access storage per DataVault at I/O bandwidths of up to 25 Mbytes/sec.
- **CM-IOPG.** This I/O controller provides 4 ports for connection to SCSI-based devices, such as cartridge tape drives.

Figure 46 shows a CM-5 system that includes a DataVault.

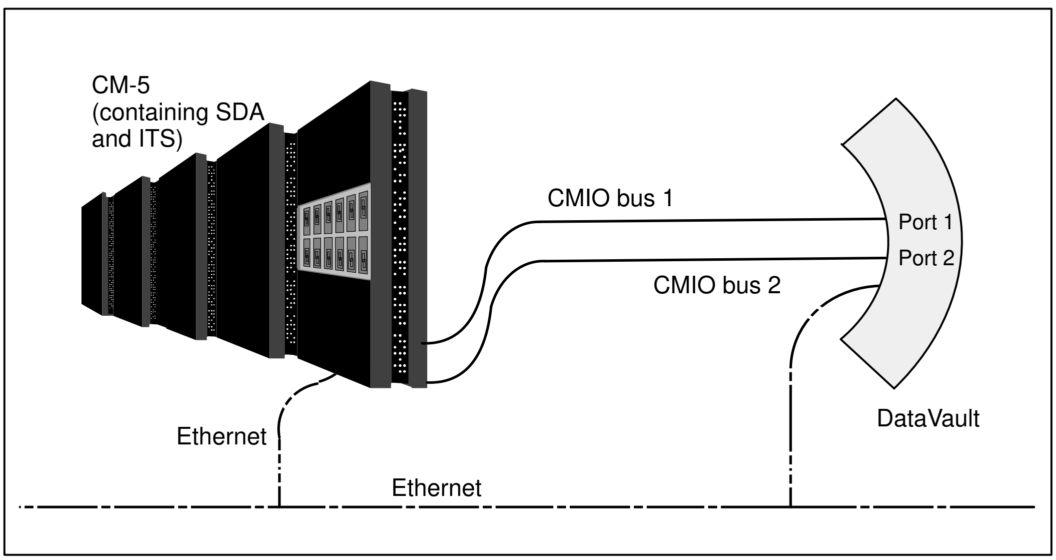

Figure 46. A sample CM I/O system for the CM-5.

The CM-5 also supports I/O for standard VMEbus and SBus devices. The VMEbus and SBus interfaces allow the CM-5 to be connected to external control processors and peripheral device controllers that implement these popular buses. These links make available to CM-5 applications a variety of other forms of I/O, including framebuffers, tertiary storage devices, and FDDI networks.

20.4.1 DataVault

The DataVault system is available in various storage capacities, ranging from 20 to 60 gigabytes. Each of these configurations is capable of transferring data at a sustained rate of 25 Mbytes/sec.

The basic DataVault storage configuration, used in the 20- and 30-gigabyte systems, employs an array of 42 $5\,^1/_4$-inch Winchester disk drives, of which 39 are active and 3 are spares. (See Figure 47.) Of the 39 active drives, 32 hold data and 7 hold error correction code (ECC) bits. The ECC bits allow the DataVault to correct single-bit errors and to flag multiple-bit errors in each 32-bit value retrieved from the disks.

The double-capacity DataVault configuration (40 or 60 gigabytes) has 84 drives, of which 64 hold data, 14 hold ECC bits, and 6 are spares.

In all DataVault configurations, each 32-bit data word is spread across 39 data and ECC drives, one bit per drive. Each 64-bit data chunk received from the CMIO bus is first split into two 32-bit words. After verifying parity from the I/O bus, the DataVault controller adds 7 ECC bits and stores the resulting 39 bits on 39 individual drives. Subsequent failure of any one of the 39 drives does not impair reading of the data, since the ECC data allows any single-bit error to be detected *and* corrected for every data word. The ECC data permits 100% recovery of the contents of a failed disk, allowing a new copy of this data to be reconstructed and written onto a spare disk. Once this recovery is complete, the database is healed.

The File Server

All I/O transactions in a DataVault I/O interface are controlled by an IOCP running a file server process. The file server manages the DataVault's UNIX-based hierarchical directory structure, handling the allocation of physical disk space and matching file names and logical read/write requests to the physical locations of data on the DataVault disks.

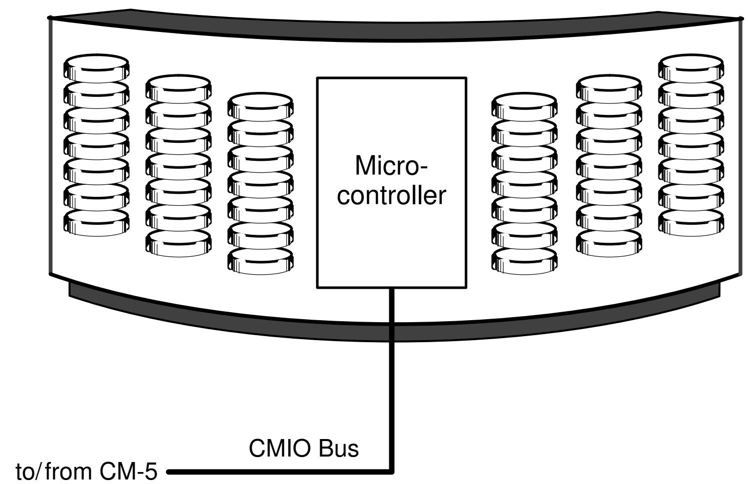

Figure 47. Inside the DataVault.

Internally, the file server represents a file as a series of *extents*, or areas of contiguous disk surface. Each extent starts at a logical offset within the file, has a physical disk address, and has a length. This representation allows a file to have arbitrarily large physically contiguous blocks of the disks holding data for logically contiguous segments of the file. As a result, positioning of the read/write heads is more efficient, yielding faster file transfer.

Writing and Reading Data

Data transfers move information between a CM-5 partition and the DataVault. The principal events involved in writing a file to the DataVault are summarized below. Reading a file from the DataVault into Connection Machine memory is very similar but the flow of data is reversed.

A DataVault write operation is typically initiated by a partition manager, which issues a write command to the IOCP that is acting as the DataVault's file server. When the file server receives the logical file request, it translates the request into a series of physical disk addresses. Assuming that the request parameters satisfy the necessary validity checks (for example, that there is sufficient space), the file server returns a message to the requesting partition indicating the DataVault's availability. If the request cannot be fulfilled, the file server returns a failure report instead.

Data from the partition's memory is moved, via the Data Network, to I/O buffers in the CMIO interface and then across the CMIO bus to the DataVault. A micro-controller within the DataVault controls the distribution of data onto the disk array. State machines at each end of the CMIO bus ensure reliable transfer of large volumes of data across the bus. Parity checking is performed on all data as it is received from the CMIO bus to ensure data integrity.

Data being read from the DataVault follows the same path as for writing, but in reverse order: across the CMIO bus, through the CMIO interface, and across the Data Network. The data coming off the disks is checked by ECC circuits. Single-bit errors are corrected and logged and the data is written with parity to the CMIO bus. As with write operations, parity checking is performed on data received from the CMIO bus.

Data Protection

The transfer status may indicate that a single disk drive is failing and that the ECC was required to correct data. This will most often be discovered when the error logs are checked. At that point, the faulty drive can be physically replaced with an external spare. If the site does not currently have any spares available in storage, other than the three (or six) spare drives contained in the DataVault, one of these internal spares can be logically substituted for the failing drive.

This logical substitution uses a software procedure, called *sparing*, that reconstructs the corrupted data, using the ECC circuits to correct the failing bit, and stores it on one of the spare drives provided for the purpose. The sparing program redirects the path followed by the faulty bit from the failing drive to the spare. Regeneration of this data takes two minutes per gigabyte, after which the data is again protected against the failure of another drive.

When the failed drive is physically replaced, the files are reconstructed using the same technique as is used when sparing the failed drive.

20.4.2 Standard Protocol I/O Interfaces

Two CM-5 standard bus interfaces, called SVME and SBA, enable the CM-5 operating system to access external VMEbus or SBus computers and their associated I/O resources using standard communications protocols. These I/O paths link the CM-5 to external networks of computing and I/O server resources.

They are connected by cable to a VMEbus- or SBus-based external control processor, which manages the file system and other I/O functions. An adapter board, installed in the control processor, provides an interface between the VMEbus and the CM-5 Data Network (and Control Network).

The SBus interface consists of an adapter board, called the SBA, that plugs into the SBus of an external control processor. This adapter board is connected by cable to an interface module, called the control processor interface (CPI), that is plugged into the CM-5 Data Network and Control Network. The external control processor, running file server code, serves as the file system processor. This arrangement allows applications running on the CM-5 to exploit any I/O resources, such as a tape storage system, that are attached to the external control processor's SBus.

In a similar fashion, the VMEbus interface uses a VME adapter board, called the SVME, to connect a control processor's VMEbus to the CM-5 Data Network and Control Network. Apart from this difference, the VMEbus interface employs the same design features as the SBus interface.