Chapter 1

Introduction to CM-5 I/O

The Connection Machine I/O system facilitates the storage, retrieval, and interprocess sharing of the massive amounts of data used and generated by the Connection Machine supercomputers. The high speed at which data transfer typically occurs significantly enhances CM application performance. The Connection Machine I/O system is both flexible and easy to use; it can incorporate a wide variety of I/O devices and access any of these devices through a variety of interfaces.

This manual describes how to use the I/O system of the Connection Machine models CM-5 and CM-5 Scale 3 to achieve typical data transfer objectives. This chapter provides background information about the CM-5 I/O system, including descriptions of

- the hardware components of the CM-5 I/O system
- the software you can use to manipulate the CM-5 I/O system and its files

This manual assumes that your CM-5 system is running Version 7.3 of the CM-5's operating system, CMOST.

1.1 I/O Components of the CM-5 System

Figure 1 and Figure 2 show the I/O hardware components of a typical CM-5 system. In addition to the CM-5 itself, the system consists of one or more I/O devices, listed below; note that several of the I/O devices reside inside the CM-5 cabinets. As shown in Figure 2, a Connection Machine model CM-2 or CM-200 may also have access to the I/O devices. Your system administrator can tell you which devices are available for general use.

Copyright © 1994 Thinking Machines Corporation

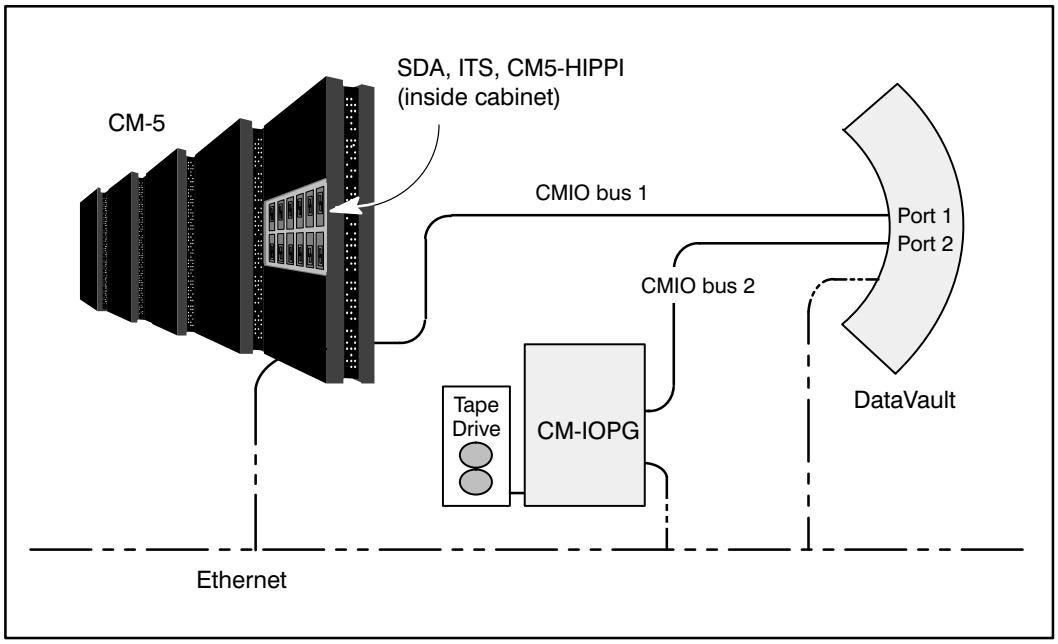

Figure 1. A typical CM I/O system for the CM-5.

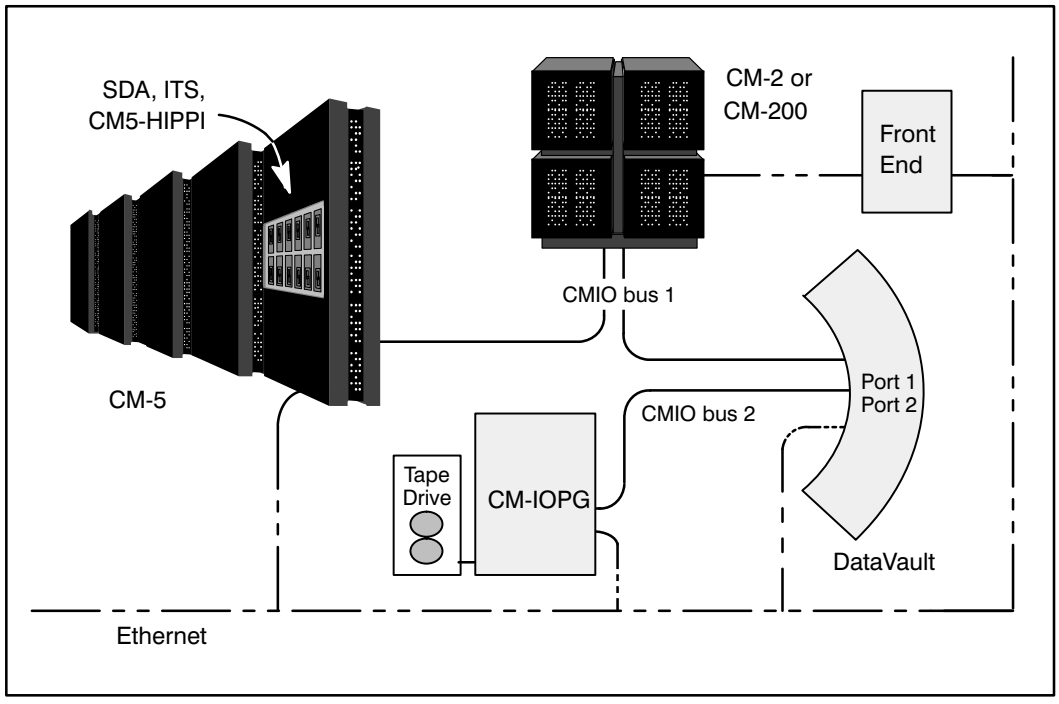

Figure 2. A typical CM I/O system shared by the CM-2 or CM-200 and the CM-5.

Copyright © 1994 Thinking Machines Corporation

As of CMOST Version 7.3, the CM-5 system supports the following I/O devices:

- Scalable Disk Array (SDA)

  The Scalable Disk Array is the CM-5 system's main storage device, typically providing 25–200 Gbytes of disk storage at I/O bandwidths of 33–264 Mbytes/second. The SDA, which resides within the CM-5 cabinets, is a collection of sets of high-speed commodity disks directly connected to the CM-5 networks. The direct connection to the networks enables each set of disks to contribute not only to storage capacity but also to I/O performance; the number of the disk sets in an SDA system can be increased or decreased to achieve an I/O system matched to the performance and capacity needs of CM-5 applications.

  An SDA supports any number of separate UNIX-compatible file systems. Data is striped over all the disks in a file system, transparently to the user. Parity is checked (and single-bit errors are corrected, if necessary) on each read operation.

- Integrated Tape System (ITS)

  The CM-5 Integrated Tape System provides fast, cost-effective tape backup, archiving, and data transfer. The ITS is frequently used for staging data from tape storage onto the SDA for subsequent use by CM-5 applications. The ITS supports a variety of tape drives, including 3480-compatible and 8-mm units.

  Like the SDA, the ITS resides within the CM-5 cabinets (in fact, SDA and ITS modules reside in the same backplane). Data transfers between the ITS and SDA media are carried out directly via the CM-5 networks under control of I/O software running on the ITS; no CM-5 partition resources are involved in the transfers. Thinking Machines provides several performance-enhanced CMOST utilities for transferring files between the SDA and tape devices that use various standard tape formats. In addition, the ITS supports the use of standard SunOS tape commands.

  Although this manual describes tape I/O and provides information on the CMOST utilities that support ITS functionality, the Thinking Machines' *ITS User's Guide*, Version 7.3, provides comprehensive information on using the ITS.

Copyright © 1994 Thinking Machines Corporation

- CM5-HIPPI (not supported on Scale 3 systems)

  The CM-5's High Performance Parallel Interface (HIPPI), fully compliant with the ANSI HIPPI standard, provides a high-speed link between the CM-5 and other computing systems, sending and receiving information at peak rates approaching 100 Mbytes/second. The CMOST operating system supports both low-level HIPPI primitives, including HIPPI-FP and HIPPI-PH, and the TCP/IP protocol suite, modified for parallel processing and HIPPI. CM5-HIPPI programming is done through CMOST I/O system calls; see the manual pages in Chapter 11, and for detailed programming information, please refer to the *CM5-HIPPI User's Guide*, Version 7.3.

  The CM-5 also supports the CM-HIPPI, which resides on the CMIO bus, as described below, and has a different programming interface.

The devices listed below reside on a high-speed multidrop CMIO bus, which has a proprietary hardware interface to the CM-5 networks:

- DataVault

  The DataVault provides up to 60 Gbytes of disk storage. It is dual-ported to allow high bandwidth data transfer and system flexibility. The DataVault file server computer, a microcomputer inside the DataVault, hosts the DataVault's UNIX-like file system.

- CM-HIPPI

  The CM-HIPPI, like the CM5-HIPPI, provides a high-speed link between the CM-5 and other computing systems. Programming is done through calls to CMFS routines; see the manual pages in Chapter 10, and for detailed programming information, please refer to the *CM-HIPPI Programming Guide*, Version 6.1.

- VMEIO Host Computer

  The VMEIO host computer typically connects VME-based I/O devices, such as video frame grabbers and magnetic tape drives, to the Connection Machine systems. Such devices provide convenient means of backing up and retrieving data. The file system that resides on the VMEIO host computer treats each device as a file, and allows the host computer itself to provide secondary disk storage for files.

Copyright © 1994 Thinking Machines Corporation

- CM-IOPG

  The CM-IOPG also supports the VME protocol. Typically, the CM-IOPG connects storage devices that have SCSI drives — such as an IBM 3480−compatible tape drive — to the Connection Machine systems. As on the VMEIO host computer, the file system that resides on the CM-IOPG treats each device as a file, and allows the host computer itself to provide secondary disk storage for files.

To communicate with the CM-5 and each other, the I/O peripherals use internal networks or a combination of the networks and external buses. The SDA, ITS, and CM5-HIPPI have a direct interface to the CM-5 processors via the CM-5's Data Network. The other peripheral devices — DataVault, CM-HIPPI, VMEIO host computer, and CM-IOPG — use Thinking Machines Corporation's proprietary CMIO bus to convey data among themselves, and to and from the IOBA, the CMIO bus's interface to the processing nodes; an Ethernet conveys control and status information.

An Ethernet can also convey data among the DataVault, VMEIO host computer, and CM-IOPG in the absence of an appropriate CMIO bus connection. In Figure 1 and Figure 2, for example, since there is no CMIO bus connection between either CM and the CM-IOPG, data flowing between the CM-2 or CM-5 and the CM-IOPG would proceed over the Ethernet. Since the Ethernet is very slow compared to a CMIO bus, if a need were to arise for frequent CM-2/CM-5 $\leftrightarrow$ CM-IOPG data transfers in the systems shown in these figures, an additional CMIO bus should be installed between the CMs and the CM-IOPG.

1.2 CM I/O

1.2.1 Parallel I/O and Serial I/O

The CM-5's operating system (CMOST) supports a unified model for *parallel I/O:* moving data in parallel between many CM-5 processing nodes and I/O storage or transfer media. Under this model, parallel programs that distribute data across many nodes can use parallel I/O primitives on file, network, or tape objects the same way serial programs use I/O primitives on traditional UNIX systems.

Existing applications that perform file I/O, create network connections, or manage tapes run largely unchanged on the CM-5; you simply change the

Copyright © 1994 Thinking Machines Corporation

code that performs I/O operations on parallel structures to use CMOST parallel I/O primitives. Calls to perform *serial I/O* — data transfer between a file and serial-computer memory, or non−data transfer file operations (e.g., **creat**) or file system operations (e.g., **statfs**) — can also be included in programs that perform parallel I/O.

Connection Machine languages support CMOST parallel I/O operations through some common abstractions. The CM Fortran **READ** and **WRITE** statements, for example, can automatically perform parallel I/O on arrays through the underlying CMOST parallel I/O primitives; similarly, data in C*'s pvars is transferred in parallel via the CMFS library's **CMFS_read_file_always** and **CMFS_write_file_always** routines. Chapter 2 lists all the available interfaces — both serial and parallel — to the CM-5 file systems.

When running an application, the CM-5 automatically takes advantage of the available parallel processing resources. Yet, because data flows through its I/O interfaces in traditional serial order, using industry-standard protocols, the CM-5 provides a seamless link between its massively parallel processing power and any other system in your data center.

1.2.2 The Client-Server Model

The Connection Machine I/O system is modeled on the typical client-server relationship: a program running on a machine acting as client requests that an I/O device running a server process either send or receive data. The following CM-5 system components can act as clients:

- a partition of CM-5 processing nodes[1]
- CM-5 ITS nodes
- a CM-5 control processor
- a CM-IOPG
- a VMEIO host computer

The following Connection Machine system components can act as servers:

- an SDA
- a CM5-HIPPI

1. The nodes may operate together as one body, or separately; see the *CMMD Reference Manual* for discussion of independent node I/O.

Copyright © 1994 Thinking Machines Corporation

- a DataVault
- a CM-IOPG
- a VMEIO host computer

This model is in keeping with the serial I/O / parallel I/O model described in Section 1.2.1:

- *Parallel I/O*

  The CM-5 — specifically, one or more processing nodes — is the client, requesting a server to read to it or accept writes from it.

- *Serial I/O*

  A CM-5 partition manager or other control processor, or a serial computer (for example, a CM-IOPG or VMEIO host computer) is the client, requesting a server to read to it or to accept writes from it.

  Serial I/O also includes non-I/O file operations — such as creating, opening, seeking, and closing — and file system maintenance operations — such as making a directory and copying a file — since these operations are directed to file server control by the partition manager or other serial computer on which the program is running.

1.3 CM-5 Data Format

Data used by the CM-5 must be in serial (single-stream) order, big-endian (left-to-right) byte order and in IEEE floating-point format. Data used by the CM-5 but not generated by it must be converted to the appropriate format if necessary.

Appendix A contains more information about data format conversion in the context of sharing data between the CM-5 and the CM-2 or CM-200.

1.4 The Connection Machine File Systems

The CM-5 system can access three file systems, two of which are proprietary to Thinking Machines Corporation. The two proprietary file systems, the

Copyright © 1994 Thinking Machines Corporation

SFS and CMFS file systems, exploit the great speed and massive storage capabilities of the Connection Machine I/O systems. The other supported file system, UNIX, 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 (Scalable File System) is an enhancement of the UNIX file system, with extensions to support parallel I/O and files much larger than most UNIX implementations can accommodate. From a user's perspective, the SFS file system appears like and behaves like a UNIX file system. (Chapters 2 and 3 provide more information about the SFS file system.)

  A CM-5 program can access the SFS file system via a variety of interfaces, including the CMMD library, the CMFS library, CMOST I/O commands and system calls, C* I/O functions, CM Fortran utility library, Fortran I/O statements, and standard UNIX routines and commands; see Chapter 2.

  Each control processor can have up to 1888 files open at a time on any one SFS file system. A single process can have open up to 64 SFS file descriptors.

- CMFS (CM File System) is a UNIX-like file system that can reside on the CMIO bus data-storage devices, including the DataVault, CMIOPG, and VMEIO host computer. Like the newer CM-5 file system, SFS, the CMFS file system supports parallel I/O and very large files. A CM-5 program can access the CMFS file system via the CMFS library, most CMOST I/O commands (but *no* CMOST I/O system calls), the CMMD library, the CM Fortran utility library, Fortran I/O statements, and, if the file system is NFS-mounted, standard UNIX routines and commands[2]; see Chapter 2.

  Together, the system's control processors can have open up to 250 files at any time on any one CMFS file system. A single process can have open up to 224 CMFS file descriptors.

  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, and the CMFS file system does not support symbolic links. Appendix A provides more information about the CMFS file system.

2. Although NFS enhances usability of the CMFS file system, this release of NFS is preliminary; other interfaces may achieve better performance than UNIX commands and calls.

Copyright © 1994 Thinking Machines Corporation

- Each CM-5 control processor has some set of local UNIX file systems. Note that this manual assumes you are familiar with the UNIX file system, or that documentation that describes it is available to you.

The UNIX file system should hold the executable programs that run on the CM-5, as well as CM-5 system software and (possibly) users' private files. Although the SFS and CMFS file systems are also general-purpose file systems, they are optimized for parallel data transfers.

From a user's perspective, the CMFS file system is completely separate from the SFS and UNIX file systems; the CMFS file system has a separate directory tree and its own current working directory, and only if the CMFS file system is NFS-mounted can UNIX recognize it. The SFS file system, however, shares name space with the UNIX file system on the CP on which it is running; as such, those two file systems share a directory tree and a current working directory, and UNIX commands (**`chdir`**, for example) can operate on the SFS file system.

If your CM-5 program has access to more than one of the CM file systems, how does it know which file system to use? This question is answered in Section 1.4.1.

1.4.1 Which File System Do You Want?

In a CM-5 system that has access to more than one of the file systems described in Section 1.4, indicate the file system you want to use by setting the CM-5 environment variable **`CMFS_PATHTYPE`** in your shell or in your **`.cshrc`** file. **`CMFS_PATHTYPE`** (as well as the **`DVWD`** and **`DVHOSTNAME`** environment variables) is recognized by CM-5 languages and libraries that use the CMFS library to perform I/O, such as C* and the CM Fortran utility library.

The setting of **`CMFS_PATHTYPE`** does not affect UNIX utilities and commands, or CM languages and libraries — such as CMMD and the CM Fortran language I/O statements — that use a mechanism other than the CMFS library to perform I/O. A relative pathname passed to a UNIX, CMMD, or CM Fortran I/O statement is resolved from the current working directory (the directory from which the application is started). Unless a pathname includes a CMFS hostname and a colon (for example, **`dv1`**: in the pathname **`dv1:/project/new_data`**), the current file system is the local UNIX or UNIX-compatible file system — for example, the SDA's SFS file system.

Copyright © 1994 Thinking Machines Corporation

CMFS_PATHTYPE Syntax

CMFS_PATHTYPE { **unix** | **cmfs** | **mixed** }

unix Any pathname refers to the local UNIX or UNIX-compatible file system — the SDA's SFS file system, if your system contains one.

cmfs Any pathname refers to a CMFS file system — a DataVault, VMEIO host computer, or CM-IOPG. If your CM-5 system has access to more than one of these devices, set the CM environment variables **DVHOSTNAME** and/or **DVWD**. (Manual pages for environment variables are in Chapter 9, "CM-5 Environment Variables: Manual Pages.") If those environment variables are not set, the default CMFS file system is used (see your system administrator).

mixed Any pathname that does not contain a colon (**:**) refers to the local UNIX or UNIX-compatible file system — the SDA's SFS file system, if your system contains one. A pathname that does contain a colon refers to a CMFS file system:

- If there is a CMFS-hostname component included before the colon, that device is used.
- If there is no CMFS-hostname component, the device listed with **DVHOSTNAME** and/or **DVWD** is used. If those environment variables are not set, the default CMFS file system is used (see your system administrator).

(For file system defaulting behavior when **CMFS_PATHTYPE** is not set, please see the man page **CMFS_PATHTYPE**(7) in Chapter 9, "CM-5 Environment Variables: Manual Pages.")

For example, in the CM-5 system shown in Figure 3, suppose the following:

- You set **CMFS_PATHTYPE** to **mixed**.
- You set **DVHOSTNAME** to **dv1**.
- You set **DVWD** to **big_project**.

Copyright © 1994 Thinking Machines Corporation

If your program calls a routine to open the file **`my_data`**, the file is opened in your current working directory on the device **`sda1`**. If your program calls a routine to open the file **`:my_data`**, the file is opened in **`dv1:/big_project`**. If your program calls a routine to open the file **`iop1:my_data`**, the file is opened in **`iop1:/big_project`**.

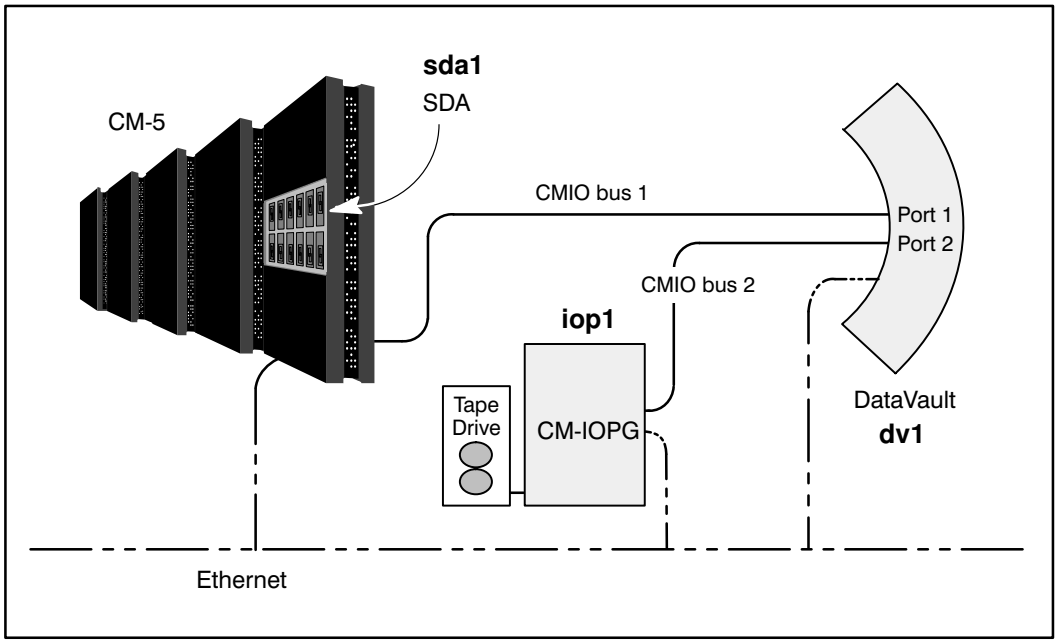

Figure 3. A typical CM I/O system for the CM-5.

Copyright © 1994 Thinking Machines Corporation