Chapter 1

Introduction

The Connection Machine CM-5 supercomputer combines high performance and ease of use for programmers working on large, complex, data-intensive applications.

This chapter begins by describing parallel programming on the CM-5, and then presents an overview of the CM-5's hardware, software, I/O, and online documentation. For a more in-depth discussion of these topics, see the *CM-5 Technical Summary*.

1.1 Parallel Programming on the CM-5

The Connection Machine CM-5 supercomputer offers users high performance through a complete range of parallel processing approaches. The CM-5 supports both data parallel programming and message-passing programming:

- The CM's data parallel compilers (CM Fortran, C*, *Lisp) present the user with a global address space and a single thread of control.
- The CMMD communications library, callable from FORTRAN 77, C, and C++ (as well as from CM Fortran and C*) provides fast communication between independent tasks (or threads of control).
- Both models utilize the rapid synchronization and low-latency communication capability of the CM-5.

In the past, programmers of supercomputers were forced to choose between these two models. The CM-5, however, supports both; in fact, data parallel

Copyright © 1993–1994 Thinking Machines Corporation

programs and message-passing programs can run simultaneously, under timesharing, on a single partition of a CM-5.

1.2 The SPMD Programming Model

It is sometimes said that virtually all successful parallel programs in existence today are written in the SPMD (Single Program, Multiple Data) style of programming: a style in which a single program runs on a multi-processor system, with each processor acting on its share of the program's multiple data items.

Both data parallel programming and message-passing programming on the CM-5 use the SPMD model. The differences are

- whether your program takes primarily a global view or a local view of the system
- the mechanisms by which the actions of the CM-5's processing nodes are coordinated

1.2.1 Data Parallel Programming

Data parallel programs take a global view of the system. A single program executes on the control processor. This program controls all the processing nodes, requesting synchronized computation, communication, and I/O as needed. These programs are written in the CM's data parallel languages, such as CM Fortran or C*, using the data parallel compilers and run-time system to control data layout and inter-processor communication and synchronization. They also make use of the CM's specialized parallel library routines and I/O functionality.

With data parallel programming, the programming nodes work in synchrony. For example, let's consider a finite difference code that needs to perform one operation on its boundary elements and another on interior

CMOST Version 7.3, April 1994
Copyright © 1993–1994 Thinking Machines Corporation

elements. Code written to deal with this case in a data parallel language would look something like

```
where (boundary_elements)
   do_a
elsewhere
   do_b
end where
```

On a CM-5, both branches can execute simultaneously. That is, each processor decides, for each data element, whether to fetch and execute the instructions for the **`where`** branch or for the **`elsewhere`** branch. All processors synchronize at the beginning and end of the **`where`** block; but within the block, each computes asynchronously with respect to the others. Most important, the synchronization is still automatic, under control of the system software, and presents no problem to the programmer.

This single flow of control, so similar to that of a standard "serial" program, makes data parallel programs the easiest of parallel programs to debug, and helps account for the popularity of this programming style.

For more information on data parallel programming, please consult the documentation for CM Fortran or C*.

1.2.2 Message-Passing Programming

Message-passing programs take a local, node-level view of the system. Again, a single program executes; but in this programming style, a separate copy of the program executes independently on each node. The nodes divide tasks and data among themselves according to the needs of the application; they may stay closely synchronized or become completely asynchronous. All communications and synchronization, as well as data layout, are under the application's explicit control.

The message-passing programming style is most useful when an application requires the dynamic allocation of tasks or data. Such applications typically use a class of algorithms known as node-expansion algorithms: examples are divide and conquer algorithms, branch and bound algorithms, asymmetric traveling salesman problems, and tree search problems.

On the CM-5, message-passing programs use the CM's communications library, CMMD. CMMD functions can be called from C, C++, FORTRAN

Copyright © 1993–1994 Thinking Machines Corporation

77, CM Fortran, or C*; the ability to use standard high-level languages is appreciated by users who have existing programs that they wish to port to a parallel supercomputer.

1.3 Scalable Computing

The ability to choose among programming models is an important feature of the CM-5, since it lets users choose the technique that is best, not only for their application, but for each part of their application. Also important is the Connection Machine's support for Scalable Computing, and its provision of tools geared specifically to the needs of its users, such as the Prism programming environment and the CMAX FORTRAN 77-to-CM Fortran translator.

Connection Machine data parallel programming has always been inherently scalable. Because the CM's data parallel software lays out data arrays at run time, a single program can run on any size Connection Machine, with computation and communication patterns optimized for machine size. The CM-5 allows message passing to be scalable as well.

1.4 CM-5 Hardware

The architecture of the CM-5 is optimized for data parallel processing of large, complex problems. Figure 1 shows this architecture.

1.4.1 Processors

A CM-5 has many parallel processing nodes, each with its own memory. Nodes can fetch from the same address in their memories to execute the same instruction, or from individually chosen addresses to execute independent instructions.

Copyright © 1993–1994 Thinking Machines Corporation

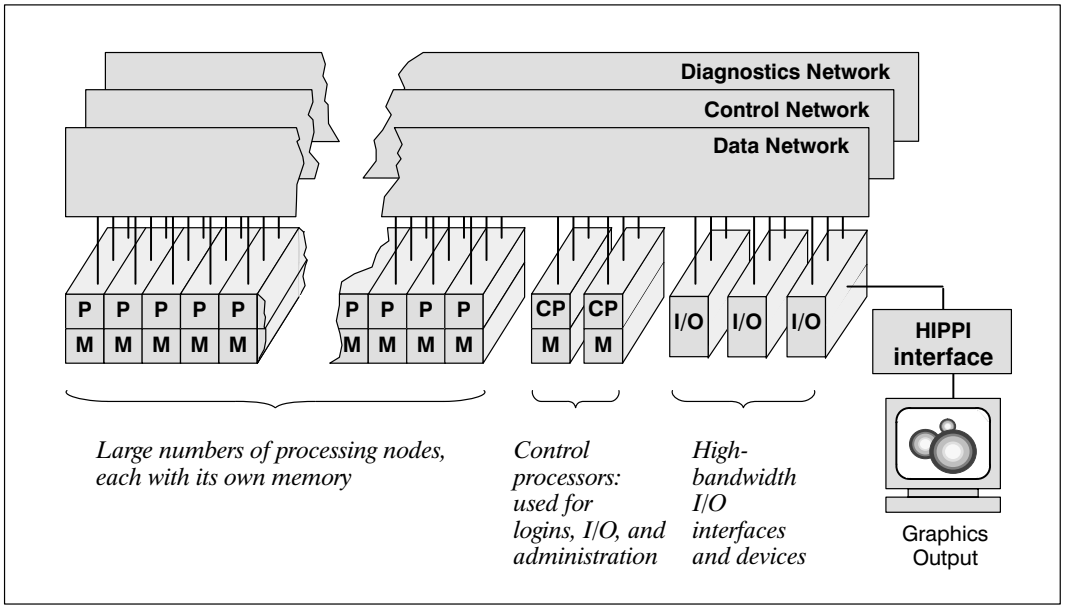

Figure 1. Organization of the CM-5.

These processing nodes are supervised by a *control processor*. The system administrator can divide the nodes into groups, known as *partitions*. There is a separate control processor, known as a *partition manager*, for each partition. Each user process executes on a single partition, but can exchange data with processes on other partitions.

Control processors that don't manage partitions manage the system's I/O devices and interfaces. This organization allows a process on any partition to gain access to any I/O device, and ensures that access to one device does not impede access to other devices.

1.4.2 Vector Units

The CM-5 can optionally contain high-performance arithmetic hardware, known as *vector units*. These vector units use wide data paths, deep pipelines, and large register files to improve peak computational performance. Users who have CM-5s with vector units typically compile their programs for, and run their programs on, the vector units.

Copyright © 1993–1994 Thinking Machines Corporation

1.4.3 Networks

Every control processor and parallel processing node in the CM-5 is connected to two communication networks, the *Data Network* and the *Control Network*.

In general, the Control Network is used for operations that involve all the nodes at once, such as synchronization operations and broadcasting. The Data Network is used for bulk data transfers in which each item has a single source and destination.

A third network, the Diagnostics Network, is visible only to the system administrator; it keeps tabs on the physical well-being of the system.

External networks, such as Ethernet and FDDI, can also be connected to a CM-5 via the control processors.

1.5 CM-5 Software

As suggested above, the Connection Machine CM-5 provides software to support both data parallel and message-passing programs. Figure 2 diagrams the current CM-5 software offerings.

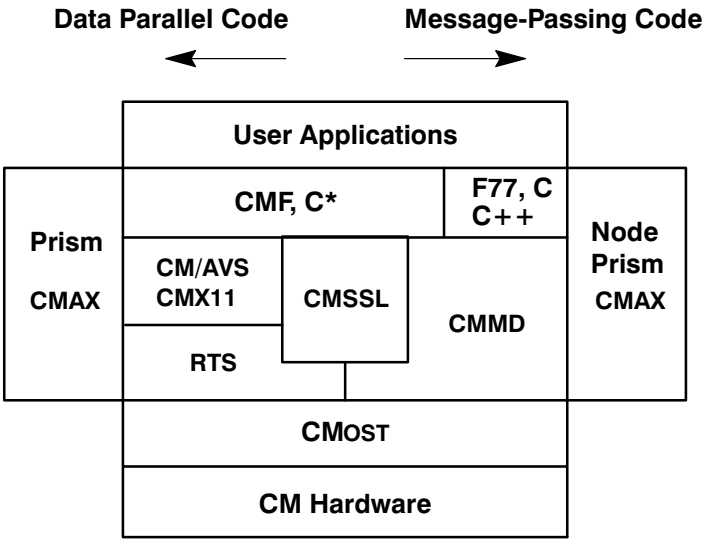

Figure 2. CM-5 software.

Copyright © 1993–1994 Thinking Machines Corporation

1.5.1 Data Parallel Software

- The CM-5 currently offers three high-level languages for data parallel programming: CM Fortran, C*, and *Lisp. See Section 4.1 for more information on CM Fortran, and Section 4.2 for more information on C*. Readers interested in *Lisp should consult the document *Porting to CM-5 *Lisp.*

- Data parallel libraries include CMSSL (the CM Scientific Software Library) and CMX11 (a visualization library that provides parallel extensions to the X11 standard). Section 4.3 briefly introduces CMSSL; Section 4.4.4 introduces CMX11.

- CM/AVS is a graphical user interface for visualization that links the computing power of the CM-5 with the convenience of a specialized graphics workstation. It is described in Section 4.4.3.

1.5.2 Message-Passing Software

Message-passing programs on the CM-5 can be written in C, C++, FORTRAN 77, CM Fortran, or C*. They are supported by a message-passing library, CMMD. This library and its associated support utilities in the operating system are summarized in Section 4.5.

1.5.3 Assemblers

The assembly language for the CM-5 is DPEAC; its assembler is **`dpas`**. The CM Fortran and C* compilers compile programs into DPEAC code.

The DPEAC language is a superset of the SPARC **`as`** assembly language. Sun's C, C++, and FORTRAN 77 compilers compile programs into **`as`**.

On a CM-5, **`as`** statements can control the SPARC microprocessors in the nodes and in the partition manager; DPEAC statements can control both the SPARC and the vector units.

The **`dpas`** assembler reads DPEAC sources and outputs UNIX object (**`.o`**) files. It does this by preprocessing its source, translating the DPEAC-specific statements into SPARC assembly source, and running the resulting

Copyright © 1993–1994 Thinking Machines Corporation

text through the **as** assembler. Any SPARC assembly statements are passed unchanged to the **as** assembler.

Because the **dpas** assembler contains an excellent preprocessor for both **as** and DPEAC statements, it is sometimes very useful as a preprocessor for an **as** program.

For full information on DPEAC and **dpas**, see the *DPEAC Reference Manual*.

1.5.4 CMAX

CMAX — the "CM Automated X-lator" — assists the conversion of standard FORTRAN 77 into CM Fortran. CMAX provides a convenient migration path for serial programs onto the CM-5. Software may be maintained in either FORTRAN 77 or CM Fortran. CMAX is described in Section 4.6.

1.5.5 The CMOST Timeshared Operating System

The CM-5 operating system, CMOST, is an enhanced version of the UNIX operating system. As such, it enables the CM-5 to interact efficiently with other devices in a heterogeneous, networked computing environment, while at the same time managing the interactions of CM-5 system components and managing the timeshared execution of multiple parallel programs.

As a resource on a heterogeneous computing network, CMOST provides

- standard UNIX utilities, user interfaces, protection, and security
- support for all standard UNIX-based communications protocols
- the ability to access files and exchange data with other systems

As a resource within the CM-5 system, CMOST provides

- fast interprocessor communications
- the parallel operations required for best utilization of processing hardware, especially the array of processing nodes

Copyright © 1993–1994 Thinking Machines Corporation

- central administration and resource management to support all CM-5 computation and I/O facilities
- timeshared execution of parallel programs

1.5.6 Prism Programming Environment

Prism is a programming environment that integrates debugging, profiling, and other useful tools in a convenient windowed environment. A graphical interface is available from terminals and workstations that are running the X Window System. A command interface is available from other terminals. You can use Prism for both data parallel and message-passing programs, as well as ordinary scalar programs. Prism is described in Chapter 3.

1.6 I/O on the CM-5

1.6.1 Hardware

The CM-5 supports a wide range of I/O devices, including:

- *Scalable Disk Array (SDA)*. The SDA is an extremely high performance, highly expandable disk storage system packaged within the CM-5 cabinetry.
- *ITS (Integrated Tape System)*. The ITS provides fast, cost-effective tape backup and archiving of SDA files. The ITS is also used for staging data from tape storage onto the SDA for subsequent use by CM-5 applications.
- *CM 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.
- *CM-2 and CM-200 I/O devices.* For application portability, the family of CM-2 and CM-200 peripherals continues to be supported on the CM-5. These devices reside on Thinking Machines' proprietary

Copyright © 1993–1994 Thinking Machines Corporation

CMIO bus, which is connected to the Data Network. These peripherals are

- *DataVault*, a high-performance, disk-based mass storage system
- *CM-IOPG*, an I/O controller that provides four ports for connection to SCSI-based devices, such as cartridge tape drives

1.6.2 Software

CM-5 software supports three file systems:

- SFS, the Scalable File System, provides access to the CM-5's high-performance Scalable Disk Array (SDA).
- CMFS, the Connection Machine File System, provides access to all I/O devices that are shared by the CM-5, CM-2, and CM-200. These devices (sometimes called the CMIO bus devices) include the DataVault mass storage system, the CM-HIPPI, and VME-based devices.
- The standard NFS-mounted UNIX file system.

The CM Fortran utility library, C* I/O procedures, and CMMD I/O routines allow applications to access disk files in any of these file systems. These procedures allow applications to open, close, truncate, and seek within files. They also allow applications to read and write data in parallel streams between the processing nodes and the SDA or DataVault.

In addition, UNIX files can often be accessed by standard UNIX commands, while special CMFS library procedures enable you to manipulate files stored on devices accessible via VME or CM-HIPPI.

This flexibility in file access, together with the fact that the CM-5 writes files in canonical UNIX order, allows the CM-5 to share data with other machines in a heterogeneous environment.

Copyright © 1993–1994 Thinking Machines Corporation

1.7 CM-5 On-Line Documentation

The CM-5 provides a hyperlinked, searchable collection of its user documentation. This on-line documentation system, called CMview, is available via Prism, or as a stand-alone application running under X.

In addition, the CM-5 provides

- UNIX-style man pages for library functions and CMOST commands
- PostScript versions of CM-5 user manuals

For further information, see Section 2.8.

Copyright © 1993–1994 Thinking Machines Corporation