Chapter 1

Introduction

This user's guide introduces the CMMD library and describes the CMOST operating system and associated utilities.

CMMD is a software library designed for message passing between the nodes or between the host and nodes on a CM-5 system. It is intended for programmers interested in node-level, message-passing programming on the CM-5 supercomputer.

This chapter briefly

- describes the CMMD programming models
- introduces the components and environment of the CM-5 system as it is used for message passing
- provides information for porting a message-passing program from other systems

1.1 CMMD Programming Models

There are two current models for message-passing programming on the CM-5 system:

- The *hostless* model: The user writes a single node program that executes in multiple copies on the processing nodes, while the host runs a CMMD-supplied I/O server program.

  This programming model is currently supported for the C, C++, C*, Fortran 77, and CM Fortran programming languages.

Copyright © 1995 Thinking Machines Corporation

- The *host/node* model: The user writes a *host program* that executes on a CM-5 control processor and cooperates with a *node program* that runs in multiple copies on the processing nodes.

  This programming model is currently supported for all the same languages as the hostless model on the nodes, but only for C, C++, and Fortran 77 on the host. The host program may not be written in C* or CM Fortran.

In both programming models, the number of processing nodes is determined by the size of the CM-5 partition you are using (see Section 1.2.1, below).

The two programming models are described in greater detail in Chapter 2.

1.2 Introducing the CM-5 System

A CM-5 is a massively parallel, scalable supercomputer that contains tens, hundreds, or thousands of processors. These processors are of two types, depending on their function: processing nodes and control processors.

- *Processing nodes* (PNs) make up the vast majority of processors in a CM-5 system. They are the processors that do the actual computation on parallel data, communicating with each other to share data as necessary. (System software occasionally refers to these processors as processing elements, or PEs.)

  On a CM-5 with vector-unit accelerators, the four VUs associated with each node are considered part of the processing node.

- *Control processors* (CPs) manage the CM-5's processing nodes and I/O devices. They provide major OS services for the system, handling the system's user interface, its I/O and network interfaces, and its system administration and diagnostic interfaces.

1.2.1 Partitions

A group of processors under the control of a single control processor is called a *partition*. The control processor that manages the partition is known as the *partition manager* (PM). In the host/node message-passing model, the

Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation

PM is the host, while the processing nodes are, of course, the nodes. In the hostless programming model, the PM merely initiates execution of the node program, and thereafter acts as an I/O server for the nodes.

Interprocessor communication networks connect all processors, of both types, to provide rapid, high-bandwidth communication between processors.

1.2.2 Operating System: CMOST

The CM-5 operating system, CMOST, is an enhanced UNIX operating system. It provides for both space-sharing and timesharing among user processes:

- Space-sharing occurs when the system administrator partitions the CM-5, allotting so many nodes to one partition, so many to another. The system administrator also decides which users have access to which partitions.

  System administrators can change partition sizes or access rules as needed to meet changing needs at their sites.

- Timesharing is the natural mode on all partitions.

  The system administrator, however, can reserve any partition for the use of one job at a time. Generally, this is done by using the Distributed Job Manager (DJM) to put the partition in *dedicated mode*, available via DJM access only (batch or interactive). See the manual *Using DJM on the CM-5* for details.

Users of the CM-5 have access to all UNIX facilities that ordinarily form part of the SunOS version of UNIX. In addition, they have access to special tools and utilities provided by CM software to facilitate parallel programming.

1.2.3 CM-5 Vector Units

In a CM-5 with vector-unit accelerators (VUs) installed, each processing node has four associated VUs. The VUs provide enhanced performance for

Copyright © 1995 Thinking Machines Corporation

parallel programs that are largely arithmetic in nature. With VUs installed, you have two options:

- You can run your code without the VUs in any of the supported serial languages (C, C++, Fortran 77).
- You can run your code with the VUs in the parallel programming languages (C* or CM Fortran) or by calling DPEAC routines from C or Fortran 77. See the *VU Programmer's Handbook* for information on DPEAC.

1.3 A Note to Users from Other Systems

If you are porting message-passing programs from another system, you may notice several differences between its environment and the CM environment:

- The CM-5 may well contain more nodes than any other machine for which you've written a message-passing program. Furthermore, it will let you write a message-passing program that uses every one of them.

  Programs ported from other machines may, therefore, spend different proportions of time computing and communicating. A program that ran on 8 or 16 nodes, spending most of its time doing computation, may find, when it runs on a considerably larger number of nodes, that its computation time is decreased and that its total time is much more evenly balanced between computation and communication. In some cases, this may mean it's a good idea to rethink your algorithms.

- You don't have to compete with other processes for nodes; the PMs run timesharing, so all partitions are timeshared. (The PM is the controlling processor here; when a host program that runs on the PM is swapped in or out, all its node programs are swapped in or out as well — host and node programs are treated as a single unit insofar as timesharing is concerned.)

- On a CM-5, you don't need to allocate processors for your program: once you access a PM (via `rlogin`, `rsh`, or a batch command), you have automatically allocated all the nodes on that PM's partition.

Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation

Therefore, your programs will always have nodes available to them, but they should plan on using all the available nodes.

- The size of any given partition is configured by the system administrator; partition sizes may therefore change from day to day, or even from hour to hour. A PM that controls 128 nodes today may control 512 nodes tomorrow.

  You can, however, count on a partition size being a power of two: 32, 64, 128, 256, 512, etc. This may be of some help to you in planning your program. (For instance, it facilitates the use of binary trees within programs.)

  Because of this flexible partitioning, your programs should make layout decisions at run time, so as to make optimal use of partitions of various sizes. You can use the `cmps` command (at shell level) and the `CMMD_partition_size` routine (within a message-passing program) to find out the current partition size. See Chapter 4.

- You will want to use CMMD's global routines whenever possible, rather than hand-code your own such routines, because CMMD's global routines take advantage of hardware and software designed specifically for best performance when operating over all nodes in a partition. (You will need to write your own routines if you want to do "global" operations, such as reduction, on a subset of the nodes in the partition.)

- CMMD is designed to save space and time by minimizing buffering. Programming styles that involve large numbers of sends from a given node followed by large numbers of receives are not appropriate for CMMD. For more details on programming strategy, see Part I of the *CMMD Reference Manual* for CMMD Version 3.3.

Copyright © 1995 Thinking Machines Corporation