Chapter 4

Executing Programs

The program execution environment on the CM-5 is similar to that of any UNIX system, with enhancements to handle parallel processing. This chapter covers the steps in running your program:

- gaining access to a CM-5
- checking system status
- executing a program
- timing a program

The last section of this chapter presents a table of useful commands and tools.

4.1 Gaining Access

To gain access to a CM-5, you must know the name of one of its partition managers, and you must have been granted access to it by the system administrator.

The CM-5 is usually accessed across a network, either by logging in remotely (via the UNIX `rlogin` command), by running a remote shell (via the `rsh` command), or by submitting a batch job (via the `jsub` or `qsub` command).

Copyright © 1995 Thinking Machines Corporation

Once you have logged in or established your shell, you are operating in the CMOST timesharing environment, with the following resources available to you:

- A partition manager (equivalent to a UNIX workstation). You initiate program execution on this processor, which uses parallel nodes and I/O devices as needed.
- All the parallel nodes in the partition. Under the CMOST timesharing environment, all the nodes are available to, and are used by, all the parallel programs running on that partition.
- All the I/O devices on the CM-5 (assuming the system administrator has granted you access to the appropriate file systems).

Figure 1 illustrates a sample CM-5 system as it appears to a user. This particular system has two partition managers, named Mars and Venus, each of which is currently managing a partition of 256 nodes. The system also has control processors managing some I/O peripherals, and one that is dedicated as a system console, for the system administrator's use.

Users (shown here at workstations "somewhere on the network") access the CM-5 system by accessing one of the PMs. They can log in remotely or use remote shells to run programs on either partition, assuming they have been granted access. An example:

```
% rlogin mars
  <login sequence>
% a.out
```

or:

```
% rsh venus my_program
```


Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation

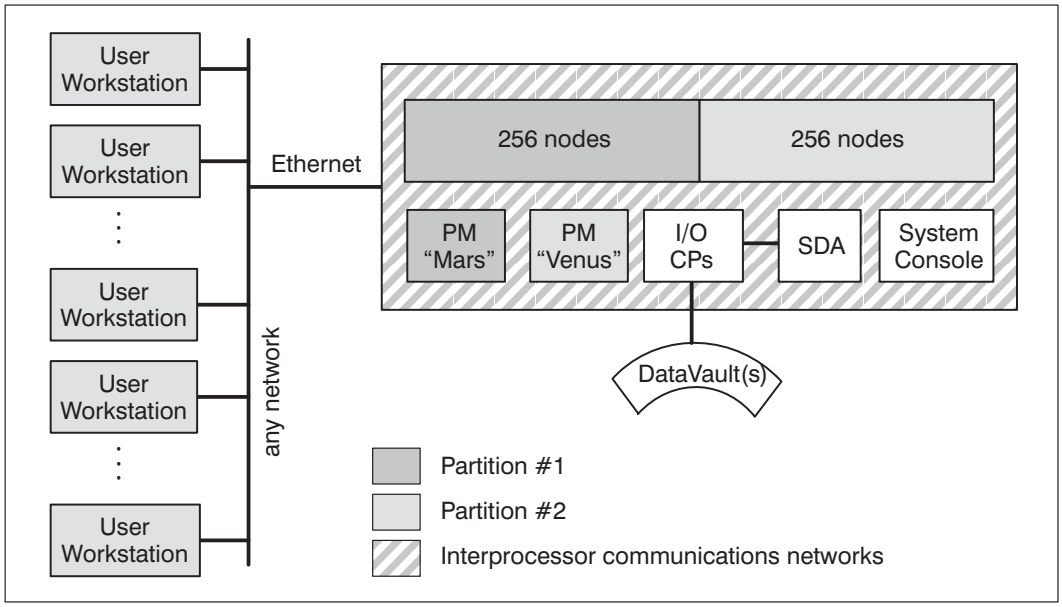

Figure 1. A sample CM-5 system.

A program runs on a single partition, using all the nodes on the partition for its parallel operations. If a program needs access outside the partition — to read from an I/O device, for example, or to pass data to a process running on another partition — it goes through the partition manager to do so. (The partition manager, running in supervisor mode, can access any address in the system. The nodes, running the user's program in user mode, can access only addresses within their own partition.)

4.2 Checking System Status

The two most common questions about system status on a CM-5 are

- How large is this partition at this time?
- How many users are running on it?

You can use the `cmps` command (see Section 4.5) to answer these questions. The `cmps` command provides information about the partition on which the command runs. If you're logged on to Mars, the command `cmps` provides information about Mars. To find out about conditions on Venus, you would use a remote shell and type `rsh venus cmps`.

Copyright © 1995 Thinking Machines Corporation

In either case, the cmps output would look something like this:

```
% cmps

32 PN System, 29168K mem. free, 6160K VU mem. free, 2 procs,
TS-9/13/92-23:00 Up: 2 day, 2:57

USER  PID  CMPID  TIME   TEXT  ILH   ILS  IGS  IGH  VUS  VUH  COMMAND

mob   *900   1   234:13  896K  254K  56K   OK   OK  64K  8K    primes
rhv   7900   0    16:36  200K  230K  56K   0K   0K  64K  0K    wil
```

The first line of the cmps output provides general information about the partition, including the number of nodes (or PNs) it contains and the amount of memory available on the nodes and the vector units.

The columns give information about each process:

- The PID and CMPID fields give the process's SunOS and CMOST process-ID number.
- The TIME field indicates the amount of time that the CMOST time-sharing daemon has made available to the process, regardless of whether the process actually used the nodes.
- The TEXT field indicates the total amount of program memory allocated to the process.
- The fields ILH, ILS, IGS, IGH, VUS, and VUH give specifics on stack and heap memory used by the process. Note these points:
  - The numbers reported are the maximum for any single node or (in the case of VUS and VUH), any single vector unit. Thus, in the case of VUS and VUH, the total amount of memory taken up per node is 4 times the total of these two fields.
  - Memory is allocated uniformly across the nodes, even in message-passing processes.
  - To calculate the total amount of memory used by a process, you must multiply by the number of nodes or vector units in the partition on which it is running.

  For more details on these fields, see the cmps man page on-line.

Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation

- The `COMMAND` field gives the name of the program being executed.

To find comparable data for the partition manager, use the UNIX `ps` command.

4.3 Executing a Program

The CMOST operating system treats the partition manager and its nodes as a single unit. Thus, you execute a message-passing program, or other parallel program, simply by executing the host program on the PM, as you would any UNIX program on any UNIX system:

```
% a.out
```

You can also execute a program in the background:

- via DJM's `jsub` or `jrun` command, if your system runs the Distributed Job Manager (DJM)
- via NQS's `qsub` command, if your system runs the Network Queueing System (NQS)
- or via the `at` or `batch` command, as on any UNIX system

See *Using DJM on the CM-5* for information about executing interactive or batch programs from within DJM. See *NQS for the CM-5* for information on executing batch programs from within NQS.

4.4 Timing a Program

If you want to time your message-passing program, you insert calls to the CMMD timers within the program. These timers are much like the CM timers used to time data parallel code, but where those timers treat all the nodes as a unit, the CMMD timers treat each node separately. Each node can have up to 64 timers active (identified by integers from 0 to 63). Each node's timers record times for only that node.

Copyright © 1995 Thinking Machines Corporation

Note that these timers execute on the nodes only. CMMD no longer offers timers that execute on the host.

Timers measure three values:

- *Busy time* is the time during which the user program is executing user code.
- *Idle time* is the time during which the user program is looping in the operating system's dispatch loop. By design, this does not occur in CMMD.
- *Elapsed time* is the sum of busy time and idle time. It represents the amount of time during which the process was scheduled for execution on the CM-5. Thus, it measures time only during this program's time-slices.

Timers give most accurate results when the program being timed has exclusive use of the partition. System load under timesharing can affect program timings.

NOTE: The CMMD timers do not measure wall-clock time. The UNIX functions `gettimeofday` and `ctime`, callable on the node (or on the host), can provide this measure.

The CMMD timing functions are

```
int CMMD_node_timer_clear (int timer)
int CMMD_node_timer_start (int timer)
int CMMD_node_timer_stop (int timer)

double CMMD_node_timer_busy (int timer)
double CMMD_node_timer_elapsed (int timer)
double CMMD_node_timer_idle (int timer)
```

For more information on using timers, see the *CMMD Reference Manual*, or the on-line manual pages.

NOTE: `CMMD_node_timer_idle` and `CMMD_node_timer_elapsed` are relics related to archaic CMMD functionality that is no longer used. Basically, in any CMMD program, `CMMD_node_timer_idle` will always return zero, and `CMMD_node_timer_elapsed` will always return the same value as `CMMD_node_timer_busy`. Therefore `CMMD_node_timer_idle` and `CMMD_node_timer_elapsed` should not be used.

Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation

4.4.1 Using CMOST Timers

It is also legal to use CMOST timing routines (`CM_timer_start`, `CM_timer_clear`, `CM_timer_stop`) in your CMMD code — the two sets of timers are separate and compatible, and operate similarly. When called from CM Fortran, for example, the CMOST functions gather timing information independently on each node. You must, however, remember to include the appropriate CMOST header files for the timers you use. For example, the CM Fortran interface to the CMOST timers is given by the header file

```
'timer-fort.h'
```

See the *CM-5 User's Guide* for more information on using CMOST timers.

4.5 Commands and Tools

The table below lists commands and tools that are useful to programmers writing message-passing programs for the CM-5 system:

`rlogin`	The standard UNIX remote login command, used to log in to a CM partition.
`rsh`	The standard UNIX remote shell execution command, used to run a program interactively on a CM partition without logging in.
`cmps`	The CM version of the UNIX `ps` command. `cmps` tells you how large a partition is and what processes are currently running on it. See Section 4.2, or the on-line man page.
`cmmd-ld`	The C version of the UNIX `ld` linker, documented in Section 3.4 of this manual and in an on-line man page.
`jrun`, `jsub`	The DJM commands by which users submit jobs for interactive and batch execution, respectively, on the CM-5. See Sections 4.1 and 4.3, the on-line manual page, or *Using DJM on the CM-5*.

Copyright © 1995 Thinking Machines Corporation

`qsub`	The NQS command by which users submit jobs for batch execution on the CM-5. See Sections 4.1 and 4.3, the on-line manual page, or *NQS for the CM-5,* Version 2.0.
Prism	The CM's parallel and node-level programming environment, documented in Prism's on-line help system, and in the *Prism User's Guide* and the *Prism Reference Manual*.
CMMD timers	Node-level timers, which allow timing of code running on individual nodes. See Section 4.4, or the on-line man page for any of the timer commands.
`CM_panic` `CMPN_panic`	CMOST calls that handle errors in host and node processes; by default, they halt the program, print an error message, and dump core. See Chapter 5, or the on-line man page.
`CMMD_error`	The CMMD interface to the `CM_panic` and `CMPN_panic` functions. See Chapter 5, or the on-line man page.
core dump files, error files	Files created by CMOST to help diagnose program errors, documented in Chapter 5 of this manual.

Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation