Chapter 1

The CM-5 System

The CM Fortran compiler's target for compiled code is the parallel computing environment of the CM-5. This chapter presents a brief overview of the low-level hardware and software of the CM-5. For more detail, please refer to the *CM-5 Technical Summary*.

1.1 The CM-5 Hardware

A typical CM-5 system consists of

- A number of processing nodes (PN), used for parallel computation, and grouped together into one or more *partitions*. Each processing node has an associated bank of distributed memory.
- Two high-speed, high-bandwidth communications networks, called the *Data Network* and the *Control Network*.
- At least one *partition manager* (PM) that serves as an intermediary between a partition of processing nodes and one or more users who are connected to the CM via an external network.
- Optional I/O control processors (IOCPs) and I/O processing nodes, used to control and communicate with parallel I/O devices (such as the Scalable Disk Array, a HIPPI connection, etc.).

In addition, on CM-5s with the vector unit option installed, each PN has four vector unit (VU) accelerators, capable of performing high-speed vector arithmetic calculations. The PN's bank of memory is divided up so that each VU has its own associated region of memory.

Copyright © 1994 Thinking Machines Corporation

Note: The descriptions and examples in this document generally assume a CM-5 with vector units. Virtually all performance and programming comments given here apply equally to non-VU CM-5 systems.

Figure 10 shows the internal structure of a CM-5 system with vector units.

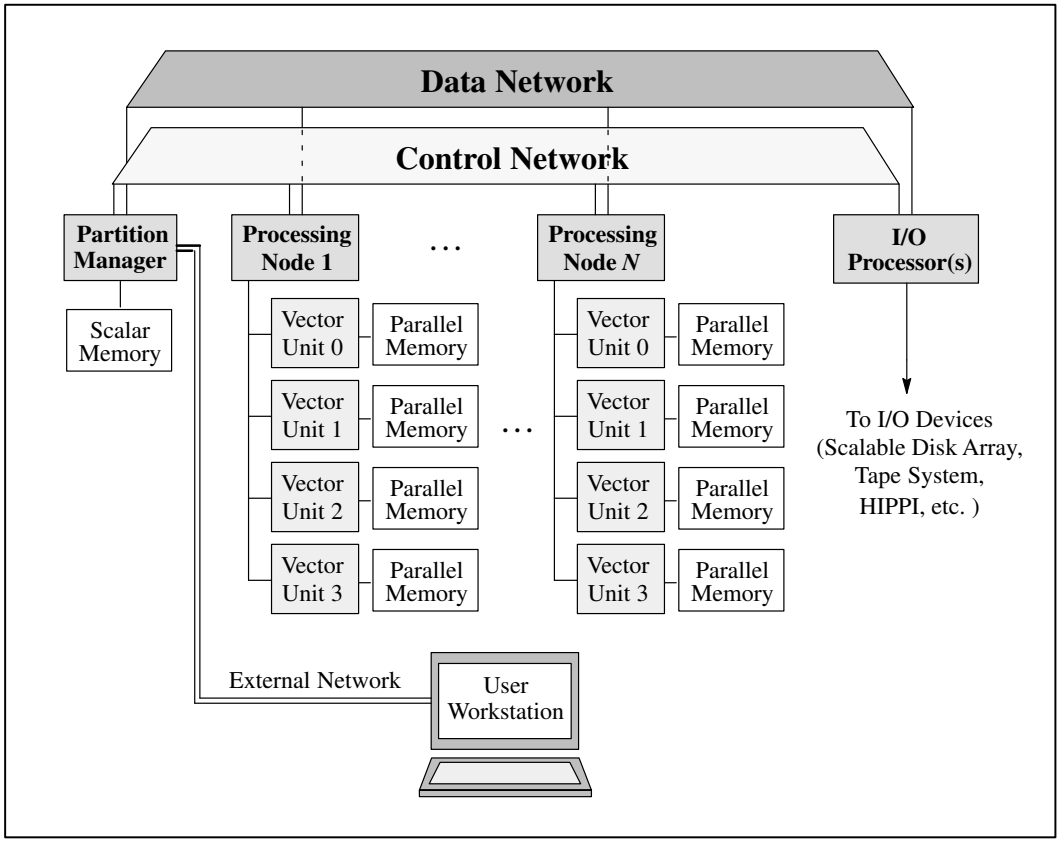

Figure 10. Components of a typical CM-5 system.

The two networks and the processing nodes are the key components of the CM-5. The nodes provide parallel computation on data stored in the CM's distributed memory. The networks provide parallel communication between nodes, distributing data where it is needed within the machine.

The Data Network is used for point-to-point exchange of messages between individual processing nodes. Currently, the Data Network transfers data at upwards of 5 Mbytes/second per node.

Version 2.1, January 1994
Copyright © 1994 Thinking Machines Corporation

The Control Network is used for operations that require all the nodes to work together, such as partitioning of the processing nodes, general broadcast of messages, and global synchronization.

1.1.1 The Processing Nodes

Each CM-5 processing node contains a RISC microprocessor, a Network Interface chip linked to the Data Network and Control Network, up to 128 Mbytes of memory, and the optional four vector units. These components are linked in a bus arrangement, as shown in Figure 11.

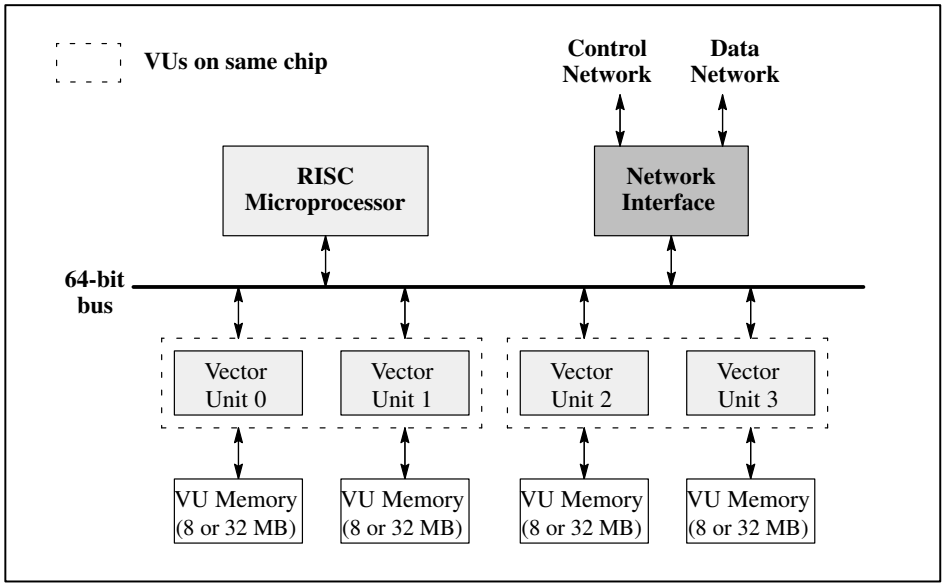

Figure 11. Components of a processing node with vector units.

1.1.2 The Partition Manager

The partition manager also contains a RISC microprocessor and a Network Interface chip. The PM is a CM user's "gateway" to the associated partition of processing nodes, and includes networking and I/O hardware for communicating with remote terminals, disk and tape drives, etc. In many respects, the PM is like an ordinary workstation that is used to communicate with the

Copyright © 1994 Thinking Machines Corporation

hardware of the CM. The PM has its own associated bank of memory, used for storage of scalar data structures and program code.

1.1.3 The Vector Units

The four vector units on a processing node act both as memory managers for the node's microprocessor and as vector arithmetic accelerators. The PN microprocessor can read or write any word in the four banks of VU memory, simply by passing an appropriate memory read/write instruction to the VUs.

If the address used in these read/write operations has certain special bits set, the memory reference is interpreted by the VUs as a vector arithmetic operation. Depending on the bits that are set, either one, two, or all four of the VUs execute a vector operation, each VU operating on data in its associated memory bank.

Note that the VUs are not independent processors; they do not load and execute instructions themselves. Instead, they respond to instructions given them by the microprocessor, providing the functions of both a memory controller and vector arithmetic accelerator. Nevertheless, since each VU has its own region of parallel memory and its own data, it is convenient to think of them as acting like processors in their own right.

1.1.4 VU Hardware and Communication Speeds

Each node has two VU chips. Each VU chip contains hardware for two vector units, so the VUs are implicitly grouped in pairs (as indicated by the dashed boxes in Figure 11). VUs 0 and 1 are on one chip, VUs 2 and 3 on the other. This affects the speed of data transfer between the memory regions of the VUs. The fastest exchanges take place between the two VUs on the same chip; the next fastest between VUs on different chips on the same node. Transfers between different processing nodes require network communication, which takes longer.

Version 2.1, January 1994
Copyright © 1994 Thinking Machines Corporation

1.1.5 Processing Element (PE)

The term *processing element*, or PE, is often used to describe the smallest computational element on the CM-5. On a machine with vector units, each VU is considered a processing element. On a machine without VUs, each PN is a processing element. Thus, a CM-5 with vector units effectively has four times as many PEs as a CM-5 without VUs.

1.1.6 Parallel Memory

"Parallel memory" on the processing nodes is simply a region of memory allocated in each of the four VU memory banks, with the size and starting location of the region in each memory bank being the same. This parallel memory region is used for storing parallel array data structures.

The remaining VU memory (that is, memory that is not used for storage of parallel data) is accessible by the node microprocessor as ordinary scalar memory. Thus, each PN has memory space for both scalar and parallel data. However, the scalar memory space on each PN is typically used only for internal data and program code.

Note for Non-VU Users: On CM-5s without vector units, the memory on each PN is treated as a single unit (see Figure 12), so there is only a single "parallel memory" region on each PN. This distinction affects the way parallel data is stored on VU and non-VU systems, as we'll see later.

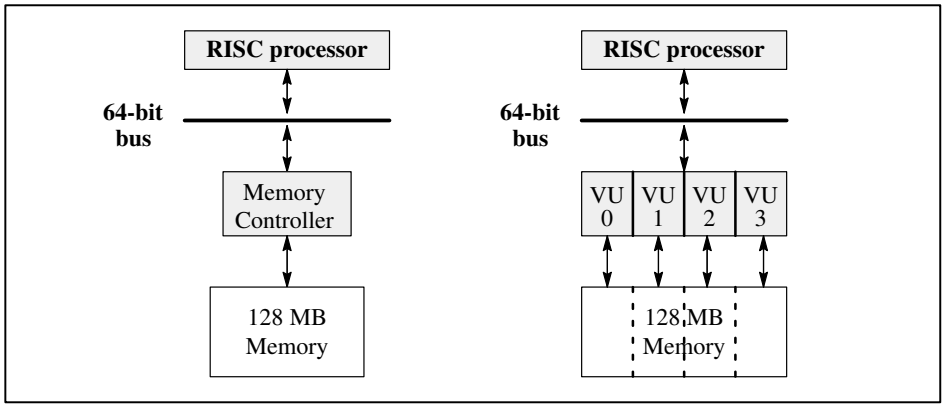

Figure 12. Assignment of node memory in non-VU and VU systems.

Copyright © 1994 Thinking Machines Corporation

1.2 Low-Level CM-5 Software

1.2.1 The CM Run-Time System

The CM Run-Time System (abbreviated CMRTS, or just RTS), is a low-level CM software layer that provides parallel memory management and interprocessor communications tools. High-level programs typically compile into a mixture of assembly language and calls to the CMRTS library. The CMRTS library provides routines for the allocation and deallocation of parallel arrays on the CM, for interprocessor exchange of array data through the CM networks, and for in-processor calculations.

The routines and data structures of the RTS are divided between the PM and the nodes. The actual array data of a parallel array is stored in PE memory. However, the *array descriptors* and other data structures that describe where the data is in CM memory are located on the PM. (There are also some low-level CMRTS data structures stored in PN memory, but these are merely conveniently accessible copies of the information on the PM.)

While user code generally doesn't call the CMRTS layer directly, an understanding of how it operates and how it keeps track of array data on the CM is useful for programmers who want to create arrays with specific layouts. We'll see some examples later on.

1.2.2 SPARC and DPEAC Assembly Code

The RISC microprocessor in the partition manager and in each PN is a SPARC chip, and thus is programmed using a standard SPARC assembly language instruction set. The VUs, however, are programmed using a separate instruction set known as *DPEAC*. The DPEAC instruction set consists of VU arithmetic and memory operations. When DPEAC code is assembled, it is converted into SPARC code consisting of memory read/write operations that send the appropriate instruction words to the VUs. Thus, DPEAC can be thought of as a convenient "shorthand" for SPARC microprocessor code that controls the VUs.

Version 2.1, January 1994
Copyright © 1994 Thinking Machines Corporation