Digital Technical Journal
Volume 6, Number 2
Cray Research paper

FOOTNOTE:

The work described in this paper was partially supported by the Advanced Research Projects Agency under Agreement No. MDA972-92-0002 dated January 21, 1992.

A Shared Memory MPP from Cray Research

by

R. Kent Koeninger, Mark Furtney, and Martin Walker

ABSTRACT

The CRAY T3D system is the first massively parallel processor from Cray Research. The implementation entailed the design of system software, hardware, languages, and tools. A study of representative applications influenced these designs. The paper focuses on the programming model, the physically distributed, logically shared memory interconnect, and the integration of Digital's DECchip 21064 Alpha AXP microprocessor in this interconnect. Additional topics include latency-hiding and synchronization hardware, libraries, operating system, and tools.

INTRODUCTION

Today's fastest scientific and engineering computers, namely supercomputers, fall into two basic categories: parallel vector processors (PVPs) and massively parallel processors (MPPs). Systems in both categories deliver tens to hundreds of billions of floating-point operations per second (GFLOPS) but have memory interconnects that differ significantly. After presenting a brief introduction on PVPs to provide a context for MPPs, this paper focuses on the design of MPPs from Cray Research.

PVPs have dominated supercomputing design since the commercial success of the CRAY-1 supercomputer in the 1970s. Modern PVPs, such as the CRAY C90 systems from Cray Research, continue to provide the highest sustained performance on a wide range of codes. As shown in Figure 1, PVPs use dozens of powerful custom vector processors on a high-bandwidth, low-latency, shared-memory interconnect. The vector processors are on one side of the interconnect with hundreds to thousands of memories on the other side. The interconnect has uniform memory access, i.e., the latency and bandwidth are uniform from all processors to any word of memory.

[Figure 1 (Memory Interconnection Architectures) in not available in ASCII format.]

MPPs implement a memory architecture that is radically different from that of PVPs. MPPs can deliver peak performance an order of magnitude faster than PVP systems but often sustain performance lower than PVPs. A major challenge in MPP design is to enable a wide range of applications to sustain performance levels higher than on PVPs.

MPPs typically use hundreds to thousands of fast commercial microprocessors with the processors and memories paired into distributed processing elements (PEs). The MPP memory interconnects have tended to be slower than the high-end PVP memory interconnects. The MPP interconnects have nonuniform memory access, i.e., the access speed (latency and bandwidth) from a processor to its local memory tends to be faster than the access speed to remote memories.

The processing speed and memory bandwidth of each microprocessor are substantially lower than those of a vector processor. Even so, the sum of the speeds of hundreds or thousands of microprocessors can often exceed the aggregate speed of dozens of vector processors by an order of magnitude. Therefore, a goal for MPP design is to raise the efficiency of hundreds of microprocessors working in parallel to a point where they perform more useful work than can be performed on the traditional PVPs. Improving the microprocessor interconnection network will broaden the spectrum of MPP applications that have faster times-to-solution than on PVPs.

A key architectural feature of the CRAY T3D system is the use of physically distributed, logically shared memory (distributed-shared memory). The memory is physically distributed in that each PE contains a processor and a local dynamic random-access memory (DRAM); accesses to local memory are faster than accesses to remote memories. The memory is shared in that any processor can read or write any word in any of the remote PEs without the assistance or knowledge of the remote processors or the operating system. Cray Research provides a shell of circuitry around the processor that allows the local processor to issue machine instructions to read remote memory locations. Distributed-shared memory is a significant advance in balancing the ratio between remote and local memory access speeds. This balance, in conjunction with new programming methods that exploit this new capability, will increase the number of applications that can run efficiently on MPPs and simplify the programming tasks.

The CRAY T3D design process followed a top-down flow. Initially, a small team of Cray Research applications specialists, software engineers, and hardware designers worked together to conduct a performance analysis of target applications. The team extracted key algorithmic performance traits and analyzed the performance sensitivity of MPP designs to these traits. This activity was accomplished with the invaluable assistance and advice of a select set of experienced MPP users, whose insights into the needs of high-performance computing profoundly affected the design. The analysis identified key fundamental operations and hardware/software features required to execute parallel programs with high performance. A series of discussions on engineering trade-offs, software reusability issues, interconnection design studies and simulations, programming model designs, and performance considerations led to the final design.

The resulting system architecture is a distributed memory, shared address space, multiple instruction, multiple data (MIMD) multiprocessor. Special latency-hiding and synchronization hardware facilitates communication and remote memory access over a fast, three-dimensional (3-D) torus interconnection network. The majority of the remote memory accesses complete in less than 1 microsecond, which is one to two orders of magnitude faster than on most other MPPs.[1,2,3]

A fundamental challenge for the CRAY T3D system (and for other MPP systems) is usability. By definition, an MPP with high usability would sustain higher performance than traditional PVP systems for a wide range of codes and would allow the programmer to achieve this high performance with a reasonable effort. Several elements in the CRAY T3D system combine to achieve this goal.

- The distributed-shared memory interconnect allows efficient, random, single-word access from any processor to any word of memory.

- Cray's distributed memory, Fortran programming model with implicit remote addressing is called CRAFT. It provides a standard, high-level interface to this hardware and reduces the effort needed to arrive at near-optimum performance for many problem domains.[4]

- The heterogeneous architecture allows problems to be distributed between an MPP and its PVP host, with the highly parallel portions on the MPP and the serial or moderately parallel portions on the PVP host. This heterogeneous capability greatly increases the range of algorithms that will work efficiently. It also enables stepwise MPP program development, which lets the programmer move code from the PVP to the MPP in stages.

- The CRAY T3D high-speed I/O capabilities provide a close coupling between the MPP and the PVP host. These capabilities sustain the thousands of megabytes per second of disk, tape, and network I/O that tend to accompany problems that run at GFLOPS.

The remainder of this paper is divided into four sections. The first section discusses the results of the applications analysis and its critical impact on the CRAY T3D design, including a summary of critical MPP functionality. The second section characterizes the system software. The software serves multiple purposes; it presents the MPP functionality to the programmer, maps the applications to the hardware, and serves as the interface to the scientist. In the third section, the hardware design is laid out in some detail, including microprocessor selection and the design issues for the Cray shell circuitry that surrounds the core microprocessor and implements the memory system, the interconnection network, and the synchronization capabilities. The fourth section presents benchmark results. A brief summary and references conclude the paper.

THE IMPACT OF APPLICATIONS ON DESIGN

As computing power increases, computer simulations increasingly use complex and irregular geometries. These simulations can involve multiple materials with differing properties. A common trend is to improve verisimilitude, i.e., the semblance of reality, through increasingly accurate mathematical descriptions of natural laws.

Consequently, the resolution of models is improving. The use of smaller grid sizes and shorter time scales resolves detail. Models that use irregular and unstructured grids to accommodate geometries may be dynamically adapted by the computer programs as the simulation evolves. The algorithms increasingly use implicit time stepping.

A naive single instruction, multiple data (SIMD) processor design cannot efficiently deal with the simulation trends and resulting model characteristics. Performing the same operation at each point of space in lockstep can be extremely wasteful. Dynamic methods are necessary to concentrate the computation where variables are changing rapidly and to minimize the computational complexity. The most general form of parallelism, MIMD, is needed. In a MIMD processor, multiple independent streams of instructions act on multiple independent data.

With these characteristics and trends in mind, the design team chose the kernels of a collection of applications to represent target applications for the CRAY T3D system. The algorithms and computational methods incorporated in these kernels were intended to span a broad set of applications, including applications that had not demonstrated good performance on existing MPPs. These kernels included seismic convolution, a partial multigrid method, matrix multiplication, transposition of multidimensional arrays, the free Lagrange method, an explicit two-dimensional Laplace solver, a conjugate gradient algorithm, and an integer sort. The design team exploited the parallelism intrinsic to these kernels by coding them in a variety of ways to reflect different demands on the underlying hardware and software. For example, the team generated different memory reference patterns ranging from local to nearest neighbor to global, with regular and irregular patterns, including hot spots. (Hot spots can occur when many processors attempt to reference a particular DRAM page simultaneously.)

To explore design trade-offs and to evaluate practical alternatives, the team ran different parallel implementations of the chosen kernel on a parameterized system-level simulator. The parameters characterized machine size, the nature of the processors, the memory system, messages and communication channels, and the communications network itself. The simulator measured rates and durations of events during execution of the kernel implementations. These measurements influenced the choices of the hardware and the programming model.

The results showed a clear relationship between the scalability of the applications and the speed of accessing the remote memories. For these algorithms to scale to run on hundreds or thousands of processors, a high-bandwidth, low-latency interprocessor interconnect was imperative. This finding led the designers to choose a distributed-shared memory, 3-D torus interconnect with very fast remote memory access speeds, as mentioned in the previous section.

The study also indicated that a special programming model would be necessary to avoid remote memory accesses when possible and to hide the memory latency for the remaining remote accesses. This finding led to the design of the CRAFT programming model, which uses hardware in the interconnect to asynchronously fetch and store data from and to remote PEs. This model helps programmers to distribute the data among the shared memories and to align the work with this distributed data. Thus, they can minimize remote references and exploit the locality of reference intrinsic to many applications.

The simulations also showed that the granularity of parallel work has a significant impact on both performance and the ease of programming. Performing work in parallel necessarily incurs a work-distribution overhead that must be amortized by the amount of work that gets done by each processor. Fine-grained parallelism eases the programming burden by allowing the programmer to avoid gathering the parallel work into large segments. As the amount of work per iteration decreases, however, the relative overhead of work distribution increases, which lowers the efficiency of doing the work in parallel. Balancing these constraints contributed to the decisions to include a variety of fast synchronization mechanisms, such as a separate synchronization network to minimize the overhead of fine-grained parallelism.

SOFTWARE

Cray Research met several times a year with a group of experienced MPP users, who indicated that software on existing MPPs was unstable and difficult to use. The users believed that Cray Research needed to provide clear mechanisms for getting to the raw power of the underlying hardware while not diverging too far from existing programming practices. The users wished to port codes from workstations, PVPs, and other MPPs. They wanted to minimize the porting effort while maximizing the resulting performance. The group indicated a strong need for stability, similar to the stability of existing CRAY Y-MP systems. They emphasized the need to preserve their software investments across generations of hardware improvements.

Reusing Stable Software

To meet these goals, Cray Research decided to reuse its existing supercomputing software where possible, to acquire existing tools from other MPPs where appropriate, and to write new software when needed. The developers designed the operating system to reuse Cray's existing UNICOS operating system, which is a superset of the standard UNIX operating system. The bulk of the operating system runs on stable PVP hosts with only microkernels running on the MPP processors. This design enabled Cray Research to quickly bring the CRAY T3D system to market. The resulting system had a minimal number of software changes and retained the maximum stability and the rich functionality of the existing UNICOS supercomputing operating system. The extensive disk, tape, and network I/O capabilities of the PVP host provide the hundreds of megabytes per second of I/O throughput required by the large MPPs. This heterogeneous operating system is called UNICOS MAX.

The support tools (editors, compilers, loaders, debuggers, performance analyzers) reside on the host and create code for execution on the MPP itself. The developers reused the existing Cray Fortran 77 (CF77) and Cray Standard C compilers, with modified front ends to support the MPP programming models and with new code generators to support the DECchip 21064 Alpha AXP microprocessors. They also reused and extended the heart of the compiling systems -- the dependency-graph-analysis and optimization module.

The CRAFT Programming Model

The CRAFT programming model extends the Fortran 77 and Fortran 90 languages to support existing popular MPP programming methods (message passing and data parallelism) and to add a new method called work sharing. The programmer can combine explicit and implicit interprocessor communication methods in one program, using techniques appropriate to each algorithm. This support for existing MPP and PVP programming paradigms eases the task of porting existing MPP and PVP codes.

The CRAFT language designers chose directives such that codes written using the CRAFT model run correctly on machines that do not support the directives. CRAFT-derived codes produce identical results on sequential machines, which ignore the CRAFT directives. Exceptions are hardware limitations (e.g., differing floating-point formats), nondeterministic behavior in the user's program (e.g., timing-dependent logic), and the use of MPP-specific intrinsic functions (i.e., intrinsics not available on the sequential machines).

A message-passing library and a shared memory access library (SMAL) provide interfaces for explicit interprocessor communication. The message-passing library is Parallel Virtual Machine (PVM), a public domain set of portable message-passing primitives developed at the Oak Ridge National Laboratory and the University of Tennessee.[5] The widely used PVM is currently available on all Cray systems. SMAL provides a function call interface to the distributed-shared memory hardware. This provides a simple interface to the programmer for shared memory access to any word of memory in the global address space. These two methods provide a high degree of control over the communication but require a significant programming effort; a programmer must code each communication explicitly.

The CRAFT model supports implicit data-parallel programming with Fortran 90 array constructs and intrinsics. Programmers often prefer this style when developing code on SIMD MPPs.

The CRAFT model provides an additional implicit programming method called work sharing. This method simplifies the task of distributing the data and work across the PEs. Programmers need not explicitly state which processors will have which specific parts of a distributed data array. Similarly, they need not specify which PEs will perform which parts of the work. Instead, they use high-level mechanisms to distribute the data and to assist the compiler in aligning the work with the data. This technique allows the programmers to maximize the locality of reference with minimum effort.

In work sharing, programmers use the SHARED directives to block the data across the distributed memories. They distribute work by placing DO SHARED directives in front of DO loops or by using Fortran 90 array statements. The compiler aligns the work with the data and doles out each iteration of a loop to the PE where most of the data associated with the work resides. Not all data needs to be local to the processor.

The hardware and the programming model can accommodate communication-intensive programs. The compiler attempts to prefetch data that resides in remote PEs, i.e., it tends to copy remote data to local temporaries before the data is needed. By prefetching multiple individual words over the fast interconnect, the compiler can mask the latency of remote memory references. Thus, locality of reference, although still important, is less imperative than on traditional MPP systems. The ability to fetch individual words provides a very fine-grained communication capability that supports random or strided access to remote memories.

The programming model is built on concepts that are also available in Fortran D, Vienna Fortran, and the proposed High-performance Fortran (HPF) language definition.[6,7,8] (Cray Research participates in the HPF Forums.) These models are based on Mehrotra's original Kali language definition and on some concepts introduced for the ILLIAC IV parallel computer by Millstein.[9,10]

Libraries

Libraries for MPP systems can be considered to consist of two parts: (1) the system support libraries for I/O, memory allocation, stack management, mathematical functions (e.g., SIN and COS), etc., and (2) the scientific libraries for Basic Linear Algebra Subroutines (BLAS), real and complex fast Fourier transforms, dense matrix routines, structured sparse matrix routines, and convolution routines. Cray Research used its current expertise in these areas, plus some third-party libraries, to develop high-performance MPP libraries with all these capabilities.

Tools

A wide variety of support tools is available to aid application developers working on the CRAY T3D system. Included in the Cray tool set are loaders, simulators, an advanced emulation environment, a full-featured MPP debugger, and tools that support high-level performance tuning.

Performance Analysis. A key software tool is the MPP Apprentice, a performance analysis tool based in part on ideas developed by Cray Research for its ATExpert tool.[11] The MPP Apprentice tool has expert system capabilities to guide users in evaluating their data and work distributions and in suggesting ways to enhance the overall algorithm, application, and program performance.

The MPP Apprentice processes compiler and run-time data and provides graphical displays that relate performance characteristics to a particular subprogram, code block, and line in the user's original source code. The user can select a code block and obtain many different kinds of detailed information. Specific information on the amount of each type of overhead, such as synchronization constructs and communication time, let the user know precisely how and where time is being spent. The user can see exactly how many floating-point instructions, global memory references, or other types of instructions occur in a selected code block.

Debugging. Cray Research supplies the Cray TotalView tool, a window-oriented multiprocessor symbolic debugger based on the TotalView product from Bolt Beranek and Newman Inc. The Cray TotalView tool is capable of debugging multiple-process, multiple-processor programs, as well as single-process programs, and provides a large repertoire of features for debugging programs written in Fortran, C, or assembly language.

An important feature of the debugger is its window-oriented presentation of information. Besides displaying information, the interface allows the user to edit information and take other actions, such as modifying the values of the variables.

The debugger offers the following full range of functions for controlling processes:

- Set and clear breakpoints (at the source or machine level)
- Set and clear conditional breakpoints and evaluation points
- Start, stop, resume, delete, and restart processes
- Attach to existing processes
- Examine core files
- Single-step source lines through a program, including stepping across function calls

Emulator. Cray Research has implemented an emulator that allows the user to execute MPP programs before gaining access to a CRAY T3D system by emulating CRAY T3D codes on any CRAY Y-MP system. The emulator supports Fortran programs that use the CRAFT model, including message-passing and data-parallel constructs, and C programs that use message passing. Because it provides feedback on data locality, work distribution, program correctness, and performance comparisons, the emulator is useful for porting and developing new codes for the CRAY T3D system.

HARDWARE

A macro- and microarchitecture design was chosen to resolve the conflict of maximizing hardware performance improvements between generations of MPPs while preserving software investments. This architecture allows Cray Research to choose the fastest microprocessor for each generation of Cray MPPs. The macroarchitecture implements the memory system and the interconnection network with a set of Cray proprietary chips (shell circuitry) that supports switching, synchronization, latency-hiding, and communication capabilities. The macroarchitecture will undergo only modest changes over a three-generation life cycle of the design. Source code compatibility will be maintained. The microarchitecture will allow the instruction set to change while preserving the macroarchitecture.

Macroarchitecture

The CRAY T3D macroarchitecture has characteristics that are both visible and available to the programmer. These characteristics include

- o Distributed memory
- o Global address space
- o Fast barrier synchronization, e.g., forcing all processors to wait at the end of a loop until all other processors have reached the end of the loop
- o Support for dynamic loop distribution, e.g., distributing the work in a loop across the processors in a manner that minimizes the number of remote memory references
- o Hardware messaging support
- o Support for fast memory locks

Memory Organization

The CRAY T3D system has a distributed-shared memory built from DRAM parts. Any PE can directly address any other PE's memory, within the constraints imposed by security and partitioning. The physical address of a data element in the MPP has two parts: a PE number and an offset within the PE, as shown in Figure 2.

Figure 2 Memory Layout

```
       +------------+ +------------+ +------------+       +------------+
       | OFFSET 0   | | OFFSET 0   | | OFFSET 0   |       | OFFSET 0   |
       +------------+ +------------+ +------------+       +------------+
       | OFFSET 1    | | OFFSET 1    | | OFFSET 1    |        | OFFSET 1    |
       +-------------+ +-------------+ +-------------+        +-------------+
       | OFFSET 2    | | OFFSET 2    | | OFFSET 2    |        | OFFSET 2    |
       +-------------+ +-------------+ +-------------+        +-------------+
       |      .      | |      .      | |      .      |        |      .      |
       |      .      | |      .      | |      .      | . . .  |      .      |
       |      .      | |      .      | |      .      |        |     .      |
       +-------------+ +-------------+ +-------------+        +-------------+
       | OFFSET M-1 | |OFFSET M-1   | | OFFSET M-1 |        | OFFSET M-1 |
       +-------------+ +-------------+ +-------------+        +-------------+
            PE 0          PE 1            PE 2                    PE N-1

       KEY:
       PE  PROCESSING ELEMENT
       M   NUMBER OF WORDS PER PROCESSING ELEMENT
       N   NUMBER OF PROCESSING ELEMENTS
```

CRAY T3D memory is distributed among the PEs. Each processor has a favored low-latency, high-bandwidth path to its local memory and a longer-latency, lower-bandwidth path to memory associated with other processors (referred to as remote or global memory).

Data Cache. The data cache resident on Digital's DECchip 21064 Alpha AXP microprocessor is a write-through, direct-mapped, read-allocate cache. CRAY T3D hardware does not automatically maintain the coherence of the data cache relative to remote memory. The CRAFT programming model manages this coherence and guarantees the integrity of the data.

Local and Remote Memory. Each PE contains 16 or 64 megabytes of local DRAM with a latency of 13 to 38 clock cycles (87 to 253 nanoseconds) and a bandwidth of up to 320 megabytes per second. Remote memory is directly addressable by the processor, with a latency of 1 to 2 microseconds and a bandwidth of over 100 megabytes per second (as measured in software). All memory is directly accessible; no action is required by remote processors to formulate responses to remote requests. The total size of memory in the CRAY T3D system is the number of PEs times the size of each PE's local memory. In a typical 1,024-processor system, the total memory size would be 64 gigabytes.

3-D Torus Interconnection Network

The CRAY T3D system uses a 3-D torus for the interconnection network. A 3-D torus is a cube with the opposing faces connected. Connecting the faces provides dual paths (one clockwise and one counterclockwise) in each of the three dimensions. These redundant paths increase the resiliency of the system, increase the bandwidth, and shorten the average distance through the torus. The three dimensions keep the distances short; the length of any one dimension grows as the cube root of the number of nodes. (See Figure 3.)

[Figure 3 (CRAY T3D System) is not available in ASCII format.]

When evaluated within the constraints of real-world packaging limits and wiring capabilities, the 3-D torus provided the highest global bandwidth and lowest global latency of the many interconnection networks studied.[1,2,3] Using three dimensions was optimum for systems with hundreds or thousands of processors. Reducing the system to two dimensions would reduce hardware costs but would substantially decrease the global bandwidth, increase the network congestion, and increase the average latency. Adding a fourth dimension would add bandwidth and reduce the latency, but not enough to justify the increased cost and packaging complexity.

Network Design

The CRAY T3D network router is implemented using emitter-coupled logic (ECL) gate arrays with approximately 10,000 gates per chip. The router is dimension sliced, which results in a network node composed of three switch chips of identical design -- one each for X-, Y-, and Z-dimension routing. The router implements a dimension-order, wormhole routing algorithm with four virtual channels that avoid potential deadlocks between the torus cycle and the request and response cycles.

Every network node has two PEs. The PEs are independent, having separate memories and data paths; they share only the bandwidth of the network and the block transfer engine (described in detail later in the paper). A 1,024-PE system would therefore have a 512-node network configured as a 3-D torus with XYZ dimensions of 8 x 8 x 8.

The network moves data in packets with payload sizes of either one or four 64-bit words. Efficient transport of single-word payloads is essential for sparse or strided access to remote data, whereas the 4-word payload minimizes overhead for dense data access.

For increased fault tolerance, the CRAY T3D system also provides spare compute nodes that are used if nodes fail. There are two redundant PEs for every 128 PEs. A redundant node can be electronically switched to replace a failed compute node by rewriting the routing tag lookup table.

Latency of the switch is very low. A packet entering a switch chip requires only 1 clock cycle (6.67 nanoseconds at 150 megahertz [MHz]) to select its output path and to exit. The time spent on the physical wires is not negligible and must also be included in latency calculations. In a CRAY T3D system, all network interconnection wires are either 1 or 1.5 clock cycles long. Each hop through the network requires 1 clock cycle for the switch plus 1 to 1.5 clock cycles for the physical wire. Turning a corner is similar to routing within a dimension. The time required is 3 clock cycles: 1 clock cycle inside the first chip, 1 clock cycle for the connection between chips, and 1 clock cycle for the second chip, after which the packet is on the wires in the next dimension.

The result is an interconnection network with low latency. As stated previously in the Memory Organization subsection, the latency for a 1,024-PE system, including the hardware and software overhead, is between 1 and 2 microseconds.

Each channel into a switch chip is 16 bits wide and runs at 150 MHz, for a raw bandwidth of 300 megabytes per second. Seven channels enter and seven channels exit a network node: one channel to and one channel from the compute resource, i.e., the pair of local PEs, and six two-way connections to the nearest network neighbors in the north, south, east, west, up, and down directions. All fourteen channels are independent. For example, one packet may be traversing a node from east to west at the same time another packet is traversing the same node from west to east or north to south, etc.

The bandwidth can be measured in many ways. For example, the bandwidth through a node is 4.2 gigabytes per second (300 megabytes per second times 14). A common way to measure system bandwidth is to bisect the system and measure the bandwidth between the two resulting partitions. This bisection bandwidth for a 1,024-PE CRAY T3D torus network is 76 gigabytes per second.

Microarchitecture -- The Core Microprocessor

The CRAY T3D system employs Digital's DECchip 21064 Alpha AXP microprocessor as the core of the processing element. Among the criteria for choosing this reduced instruction set computer (RISC) microprocessor were computational performance, memory latency and bandwidth, power, schedule, vendor track record, cache size, and programmability. Table 1, the Alpha Architecture Reference Manual, and the DECchip 21064-AA Microprocessor Hardware Reference Manual provide details on the Alpha AXP microprocessor.[12,13]

Table 1 CRAY T3D Core Microprocessor Specifications

Characteristic	Specification
Microprocessor	Digital's DECchip 21064 Alpha AXP microprocessor
Clock cycle	6.67 nanoseconds
Bidirectional data bus	128 bits data, 28 check bits

Data error protection	SECDED
Address bus	34 bits
Issue rate	2 instructions/clock cycle
Internal data cache	8K bytes (256 32-byte lines)
Internal instruction cache	8K bytes (256 32-byte lines)
Latency: data cache hit	3 clock cycles
Bandwidth: data cache hit	64 bits/clock cycle
Floating-point unit	IEEE floating-point and floating-point--to--integer
Floating-point registers	32 (64 bits each)
Integer execution unit	Integer arithmetic, shift, logical, compare
Integer registers	32 (64 bits each)
Integrated circuit	CMOS, 14.1 mm x 16.8 mm
Pin count	431 (229 signal)
Typical power dissipation	-23 watts

For use in a shared address space MPP, all commercially available microprocessors contemporaneous with the DECchip 21064 device have three major weaknesses in common:[14]

1. Limited address space

2. Little or no latency-hiding capability

3. Few or no synchronization primitives

These limitations arise naturally from the desktop workstation and personal computer environments for which microprocessors have been optimized. A desktop system has a memory that is easily addressed by 32 or fewer bits. Such a system possesses a large board-level cache to reduce the number of memory references that result in the long latencies associated with DRAM. The system usually is a uniprocessor, which requires little support for multiple processor synchronization. Cray Research designed a shell of circuitry around the core DECchip 21064 Alpha AXP microprocessor in the CRAY T3D system to extend the microprocessor's capabilities in the three areas.

Address Extension

The Alpha AXP microprocessor has a 43-bit virtual address space that is translated in the on-chip data translation look-aside buffer (DTB) to a 34-bit address space that is used to address physical bytes of DRAM. Thirty-four bits can address up to 16 gigabytes (2**34 bytes). Since the CRAY T3D system has up to 128 gigabytes (2**37 bytes) of distributed-shared memory, at least 37 bits of physical address are required. In addition, several more address bits are needed to control caching and to facilitate control of the memory-mapped mechanisms that implement the external MPP shell. The CRAY T3D system uses a 32-entry register set called the DTB Annex to extend the number of physical address bits beyond the 34 provided by the microprocessor.

Shell circuitry always checks the virtual PE number. If the number matches that of the local PE, the shell performs a local memory reference instead of a remote reference.

Latency-hiding Mechanisms

As with most other microprocessors, the external interface of the DECchip 21064 is not pipelined; only one memory reference may be pending at any one time. Although merely an annoyance for local accesses, this behavior becomes a severe performance restriction for remote accesses, with their longer latencies, unless external mechanisms are added to extend the processor's memory pipeline.

The CRAY T3D system provides three mechanisms for hiding the startup time (latency) of remote references: (1) the prefetch queue, (2) the remote processor store, and (3) the block transfer engine. As shown in Table 2, each mechanism has its own strengths. The compilers, communication libraries, and operating system choose among these mechanisms according to the specific remote reference requirements. Typically, the prefetch queue and the remote processor store are the most effective mechanisms for fine-grained communication, whereas the block transfer engine is strongest for moving large blocks of data.

Table 2 Latency-hiding Attributes

	Prefetch Queue	Remote Processor Store	Block Transfer Engine
Source	Memory	Register	Memory
Destination	Local queue	Memory	Memory
Data Size	1 word	1-4 words	Up to 256K words
Startup (6.67-nanosecond clock cycles)	18-47	6-53	>480
Latency (nanoseconds)	80	40	40-80

The Prefetch Queue. The DECchip 21064 instruction set includes an operation code FETCH that permits a compiler to provide a "hint" to the hardware of upcoming memory activity. Originally, the FETCH instruction was intended to trigger a prefetch to the external secondary cache. The CRAY T3D shell hardware uses FETCH to initiate a single-word remote memory read that will fill a slot reserved by the hardware in an external prefetch queue.

The prefetch queue is first in, first out (FIFO) memory that acts as an external memory pipeline. As the processor issues each FETCH instruction, the shell hardware reserves a location in the queue for the return data and sends a memory read request packet to the remote node. When the read data returns to the requesting processor, the shell hardware writes the data into the reserved slot in the queue.

The processor retrieves data from the FIFO queue by executing a load instruction from a memory-mapped register that represents the head of the queue. If the data has not yet returned from the remote node, the processor will stall while waiting for the queue slot to be filled.

The data prefetch queue is able to store up to 16 words, that is, the processor can issue up to 16 FETCH instructions before executing any load instructions to remove (pop) the data from the head of the queue. Repeated load instructions from the memory-mapped location that addresses the head of the queue will return successive elements in the order in which they were fetched.

The Remote Processor Store. The DECchip 21064 stores to remote memory do not need to wait for a response, so a large number of store operations can be outstanding at any time. This is an effective communication mechanism when the producer of the data knows which PEs will immediately need to use the data.

The Alpha AXP microprocessor has four 4-word write buffers on chip that try to accumulate a cache line (4 words) of data before performing the actual external store. This feature increases the network packet payload size and the effective bandwidth.

The CRAY T3D system increments a counter in the PE shell circuitry each time the DECchip 21064 microprocessor issues a remote store and decrements the counter each time a write operation completes. For synchronization purposes, the processor can read this counter to determine when all of its writes have completed.

The Block Transfer Engine. The block transfer engine (BLT) is an asynchronous direct memory access controller used to redistribute data between local and remote memory. To facilitate reorganization of sparse or randomly organized data, the BLT includes scatter-gather capabilities in addition to constant strides. The BLT operates independently of the processors at a node, in essence appearing as another processor in contention for memory, data path, and switch resources. Cray Research has a patent pending for a centrifuge unit in the BLT that accelerates the address calculations in the CRAFT programming model.

The processor initiates BLT activity by storing individual request information (for example, starting address, length, and stride) in the memory-mapped control registers. The overhead associated with this setup work is noticeable (tens of microseconds), which makes the BLT most effective for large data block moves.

Synchronization

The CRAY T3D system provides hardware primitives that facilitate synchronization at various levels of granularity and support both control parallelism and data parallelism. Table 3 presents the characteristics of these synchronization primitives.

Table 3 Synchronization Primitives

Primitive	Granularity	Parallelism
Barrier	Coarse	Control
Fetch-and-increment	Medium	Both
Lightweight messaging	Medium	Both
Atomic swap	Fine	Data

Barrier. The CRAY T3D has specialized barrier hardware in the form of 16 parallel logical AND trees that permit multiple barriers to be pipelined and the resource to be partitioned. When all PEs in the partition have reached the barrier and have set the same bit to a one, the AND function is satisfied and the barrier bit in each PE's barrier register is cleared by hardware, thus signaling the processors to continue.

The barrier has a second mode, called eureka mode, that supports search operations. A eureka is simply a logical OR instead of a logical AND and can be satisfied by any one processor.

The barrier mechanism in the CRAY T3D system is quite fast. Even for the largest configuration (i.e., 2,048 PEs), a barrier propagates in less than 50 clock cycles (about 330 nanoseconds), which is roughly the latency of a local DRAM read.

Fetch and Increment. The CRAY T3D system has specialized fetch-and-increment hardware as part of a shared register set that automatically increments the contents each time the register is read. Fetch-and-increment hardware is useful for distributing control with fine granularity. For example, it can be used as a global array index, shared by multiple processors, where each processor increments the index to determine which element in an array to process next. Each element can be guaranteed to be processed exactly once, with minimal control overhead.

Messaging. A messaging facility in the CRAY T3D system enables the passing of packets of data from one processor to another without having an explicit destination address in the target PE's memory. A message is a special cache-line-size write that has as its destination a predefined queue area in the memory of the receiving PE. The shell circuitry manages the queue pointers, providing flow control mechanisms to guarantee the correct delivery of the messages. The shell circuitry interrupts the target processor after a message is stored.

Atomic Swap. Atomic swap registers are provided for the exchange of data with a memory location that may be remote. The swap is an atomic operation, that is, reading the data from the memory location and overwriting the data with the swap data from the processor is an indivisible operation. As with ordinary memory reads, swap latency can be hidden using the prefetch queue.

I/O

System I/O is performed through multiple Cray high-speed channels that connect the CRAY T3D system to a host CRAY Y-MP system or to standard Cray I/O subsystems. These channels provide hundreds of megabytes per second of throughput to the wide array of peripheral devices and networks already supported on Cray Research mainframes. Cray has demonstrated individual high-speed channels that can transfer over 100 megabytes per second in each direction, simultaneously. There are two high-speed channels for every 128 processors in a CRAY T3D system.

BENCHMARK RESULTS

The following benchmarks show results as of May 1994, six months after the release of the CRAY T3D product. The results indicate that in this short span of time, the CRAY T3D system substantially outperformed other MPPs.

As shown in Figure 4, a CRAY T3D system with 256 processors delivered the fastest execution of all eight NAS Parallel Benchmarks on any MPP of any size.[15] (The NAS Parallel Benchmarks are eight codes specified by the Numerical Aerodynamic Simulation [NAS] program at NASA/Ames Research Center. NAS chose these codes to represent common types of fluid dynamics calculations.) The CRAY T3D system scaled these benchmarks more efficiently than all other MPPs, with near linear scaling from 32 to 64, 128, and 256 processors. Other MPPs scaled the benchmarks poorly. None of these other MPPs reported all eight benchmarks scaling to 256 processors, and the scaling reported showed more nonlinear scaling than on the CRAY T3D system. These benchmark results confirm that the superior speed of the CRAY T3D interconnection network is important when scaling a wide range of algorithms to run on hundreds of processors.

[Figure 4 (NAS Parallel Benchmarks) is not available in ASCII format.]

Note that a 256-processor CRAY T3D system was the fastest MPP running the NAS Parallel Benchmarks. Even so, the CRAY C916 parallel vector processor ran six of the eight benchmarks faster than the CRAY T3D system. The CRAY T3D system (selling for about $9 million) showed better price/performance than the CRAY C916 system (selling for about $27 million). On the other hand, the CRAY C916 system showed better absolute performance. When we run these codes on a 512-processor CRAY T3D system (later this year), we expect the CRAY T3D to outperform the CRAY C916 system on six of the eight codes.

Heterogeneous benchmark results are also encouraging. We benchmarked a chemistry application, SUPERMOLECULE, that simulates an imidazole molecule on a CRAY T3D system with a CRAY Y-MP host. The application was 98 percent parallel, with 2 percent of the overall time spent in serial code (to diagonalize a matrix). We made a baseline measurement by running the program on 64 CRAY T3D processors. Quadrupling the number of processors (256 PEs) showed poor scaling -- a speedup of 1.3 times over the baseline measurement. When we moved the serial code to a CRAY Y-MP processor on the host, leaving the parallel code on 256 CRAY T3D processors, the code ran 3.3 times faster than the baseline, showing substantially more efficient scaling. Figure 5 shows SUPERMOLECULE benchmark performance results on both homogeneous and heterogeneous systems. Ninety-eight percent may sound like a high level of parallelism, but after dividing 98 percent among 256 processors, each processor ran less than 0.4 percent of the overall parallel time. The remaining serial code running on a single PE ran five times longer than the distributed parallel work, thus dominating the time to solution. Speeding up the serial code by running it on a faster vector processor brought the serial time in line with the distributed-parallel time, improving the scaling considerably.

[Figure 5 (SUPERMOLECULE Benchmark Performance Results for Homogeneous and Heterogeneous Systems) is not available in ASCII format.]

The CRAY T3D system demonstrated faster I/O throughput than any other MPP. A 256-processor system sustained over 570 megabytes per second of I/O to a disk file system residing on a solid-state device on the host. The system sustained over 360 megabytes per second to physical disks.

SUMMARY

This paper describes the design of the CRAY T3D system. Designers incorporated applications profiles and customer suggestions into the CRAFT programming model. The model permits high-performance exploitation of important computational algorithms on a massively parallel processing system. Cray Research designed the hardware based on the fundamentals of the programming model.

As of this writing, a dozen systems have shipped to customers, with results that show the system design is delivering excellent performance. The CRAY T3D system is scaling a wider range of codes to a larger number of processors and running benchmarks faster than other MPPs. The sustained I/O rates are also faster than on other MPPs. The system is performing as designed.

REFERENCES

1. R. Numrich, P. Springer, and J. Peterson, "Measurement of Communication Rates on the CRAY T3D Interprocessor Network," Proceedings HPCN Europe (Munich) (April 1994).

2. R. Kessler and J. Schwarzmeier, "CRAY T3D: A New Dimension for Cray Research," Proceedings of COMPCON (1993): 176-182.

3. S. Scott and G. Thorson, "Optimized Routing in the CRAY T3D," extended abstract for the Parallel Computing Routing and Communication Workshop (1994).

4. D. Pase, T. MacDonald, and A. Meltzer, "The CRAFT Fortran Programming Model," CRAY Internal Report (Eagan, MN: Cray Research, Inc., February 1993) and Scientific Programming (New York: John Wiley and Sons, forthcoming).

5. A. Geist et al., PVM 3 User's Guide and Reference Manual (Oak Ridge, TN: Oak Ridge National Laboratory, ORNL/TM-12187, May 1993).

6. G. Fox et al., Fortran D Language Specification (Houston, TX: Department of Computer Science, Rice University, Technical Report TR90-141, December 1990).

7. B. Chapman, P. Mehrotra, and H. Zima, Vienna Fortran -- A Fortran Language Extension for Distributed Memory Multiprocessors (Hampton, VA: ICASE, NASA Langley Research Center, 1991).

8. High Performance Fortran (High Performance Fortran Language Specification, Version 1.0) (May 1993). Also available as

technical report CRPC-TR 92225 (Houston, TX: Center for Research on Parallel Computation, Rice University) and in Scientific Computing (forthcoming).

9. P. Mehrotra, "Programming Parallel Architectures: The BLAZE Family of Languages," Proceedings of the Third SIAM Conference on Parallel Processing for Scientific Computing (December 1988): 289-299.

10. R. Millstein, "Control Structures in ILLIAC IV Fortran," Communications of the ACM, vol. 16, no. 10 (October 1973): 621-627.

11. J. Kohn and W. Williams, "ATExpert," The Journal of Parallel and Distributed Computing, 18 (June 1993): 205-222.

12. R. Sites, ed., Alpha Architecture Reference Manual (Burlington, MA: Digital Press, Order No. EY-L520E-DP, 1992).

13. DECchip 21064-AA Microprocessor Hardware Reference Manual, 1st ed. (Maynard, MA: Digital Equipment Corporation, Order No. EC-N0079-72, October 1992).

14. D. Bailey and R. Schreiber, "Problems with RISC Microprocessors for Scientific Computing" (Moffet Field, CA: NASA/Ames, RNR Technical Report, September 1992).

15. D. Bailey, E. Barszcz, L. Dagum, and H. Simon, "NAS Parallel Benchmark Results" (Moffet Field, CA: NASA/Ames, RNR Technical Report, March 1994 [updated May 1994]).

BIOGRAPHIES

R. Kent Koeninger   Kent Koeninger has been the MPP Software Program Manager for Cray Research since 1992. Prior to this, he was a supercomputer specialist for Apple Computer, where he modeled the Cray to Apple's unique interactive-graphics, mass-storage, and high-speed-networking requirements. Earlier, while at the NASA/Ames Research Center, he repeatedly upgraded the supercomputers to the fastest available. A notable event was the first field installation of the CRAY X-MP system. Kent has a B.S. (cum laude, 1977) in mathematics from California State University at Bakersfield and is a National Merit Scholar.

Mark Furtney   Mark Furtney specializes in software for high-performance parallel systems. He has been employed by Cray Research in the Software Division since 1982, where he worked on CRAY-2 parallel software and led the development of Cray's Autotasking compiling system. He is now the group leader for Tools, Libraries, Commands, and MPP Software for various systems, including the CRAY T3D and follow-on systems. Mark holds a B.S. (1968) in mechanical engineering from Clarkson University, an M.S. (1970) in nuclear engineering from MIT, and a Ph.D. (1983) in computer science from the University of Virginia.

Martin Walker   Martin Walker directed all applications activity in support of CRAY T3D development. He was co-creator and first director of Cray's Parallel Applications Technology Program. Presently, he is General Manager of APTOS, a European applications company created by Cray Research and Stern Computing Systems. Prior to joining Cray, following fifteen years of scientific research, he managed MPP development at Myrias Research Corporation. Martin has a B.Sc. from Carleton University, Ottawa, and a Ph.D. from the University of London, U.K.