Conf-910654--1

SAND--91-1026C

DE91 012825

HYDROCODE DEVELOPMENT ON THE NCUBE AND THE CONNECTION MACHINE HYPERCUBES

Allen C. ROBINSON*, Courtenay T. VAUGHAN*, H. Eliot FANG*, Carl F. DIEGERT* and Kah-Song CHO+

*Sandia National Laboratories, Albuquerque, NM 87185-5800
+Thinking Machines Corporation, Cambridge, MA 02142-1264

Hydrocode simulations constitute an important tool at Sandia National Laboratories and elsewhere for analyzing complex two- and three-dimensional systems. However, current vector supercomputers do not provide a growth path to enable fast, routine, and cost-effective simulations of large problems. Future, massively-parallel computers will provide a solution. Sandia has already developed simplified versions of the production hydrocode CTH for the Connection Machine and the nCUBE massively-parallel supercomputers. The parallel versions solve problems in two-dimensional, multi-fluid, shock-wave physics. Code development strategy, coding methodology, visualization techniques and performance results for this work are described.

1. INTRODUCTION

For many years analysts have simulated high-compression, high-deformation events in solids with continuum-mechanics codes. Originally these codes modeled one-dimensional geometries. As computer capability increased, modeling extended to two dimensions. Recently-available vector supercomputers now allow limited three-dimensional modeling. Continuum-mechanics codes typically fall into two classes, Lagrangian and Eulerian. Lagrangian codes maintain state information at grid points attached to material particles (in a continuum mechanics sense). Eulerian codes maintain information at points fixed in space. This paper reports on extending a production Eulerian code to execute on massively-parallel supercomputers.

The Eulerian code CTH[1] (developed at Sandia National Laboratories) is in production use for a wide variety of simulation purposes. Sandia's CRAY Y-MP 8/64, a vector supercomputer, can complete a typical, two-dimensional, CTH simulation in a few minutes, or a few hours, depending on the detail required in the problem. From the point of view of the analyst, who is interested in investigating various phenomena by varying modeling parameters or the physical configuration of the initial state, the longer turn-around-times are manageable but not appealing. For three-dimensional problems, the execution time for a single simulation can be a few hours to hundreds of hours with memory requirements of a few Mwords to hundreds of Mwords (1 word = 64 bits). These demands limit detailed three-dimensional simulations to those few which are extremely important and have significant financial backing to cover both the analyst's time and computing costs.

A two or three order-of-magnitude improvement in computing speed and memory capacity would make three-dimensional simulations common procedure. However, shared-memory, vector supercomputing technology (exemplified by the CRAY architecture) will be unable to provide this improvement due to fundamental limitations in achievable clock and memory access rates. Massively-parallel computing promises to provide the means to overcome these roadblocks. The basic idea is to combine a large number of relatively-inexpensive processors, and to interconnect them with fast communications paths. Extremely powerful supercomputers can be built using this simple concept.

Given that a massively-parallel machine is available, the fundamental problem remains to develop algorithms and software implementations which effectively utilize all the processors. In addition, it is necessary for the problem size to increase

*This work performed at Sandia National Laboratories supported by the U.S. DOE under contract #DE-AC04-76DP00789.

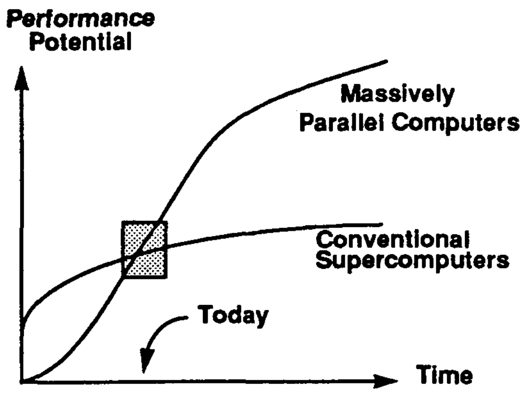

FIGURE 1
Diagram of performance potential of conventional supercomputers versus massively-parallel supercomputers.

with the number of processors so that all the processors are utilized and also to minimize the effect of inter-processor communication overhead. Fortunately, many interesting problems are large and inherently parallel. They can be efficiently mapped to a large parallel machine. Figure 1 illustrates our view of the performance potential of massively-parallel computers versus conventional supercomputers.

Sandia National Laboratories is involved in a research program to develop massively-parallel versions of the CTH hydrocode.^{2} This effort is aimed at developing coding methodologies and paradigms which will enable us to effectively utilize current and future generations of massively-parallel machines. In the process we are developing several versions of a code which we call PCTH (for Parallel CTH). Our research scope includes both the MIMD (Multiple Instruction Multiple Data) and SIMD (Single Instruction Multiple Data) distributed memory massively-parallel architectures. The MIMD architecture takes many forms and we intend to explore message passing multi-processing not only on tightly coupled MIMD machines such as the nCUBE2 hypercube produced by the nCUBE Corporation, but also on distributed workstation networks and multiple-CPU shared-memory machines. Our SIMD research is limited to the CM-2 machine produced by Thinking Machines Corporation. The remainder of this paper reviews progress in developing CTH algorithms for both the nCUBE2 and CM-2 supercomputers.

2. ARCHITECTURES

The two major target architectures for which we are building the PCTH codes are the nCUBE2 hypercube and the CM-2 which are, respectively, MIMD and SIMD machines. These machines are housed in the Massively-Parallel Computing Research Lab (MPCRL) at Sandia.

The largest nCUBE2 at Sandia is a 1024 node hypercube with 4 Mbytes of memory on each node. Each node performs at about the level of 1-2 MFLOPS (million floating point operations per second) in Fortran or C. The PCTH node code resides on each node but works on different data. The nodes run independently unless explicitly synchronized. The machine has a total of 512 Mwords of memory with a peak speed of about 2 GFLOPS. A set of 16 one Gbyte disks is available for parallel I/O. Figure 2 illustrates the PCTH configuration for the nCUBE.

The Connection Machine (CM-2) at Sandia has 16K 1-bit processors (out of a possible 64K) which have access to 512 64-bit Weitek floating point units which are the main computational units utilized for our application. Sandia's CM-2 has 256 Mwords of memory and a 2.5 Gword DataVault. Code on Sandia's 16K Connection machine runs at between one and two GFLOPS in double-precision. The SIMD machine is unique in that the executing

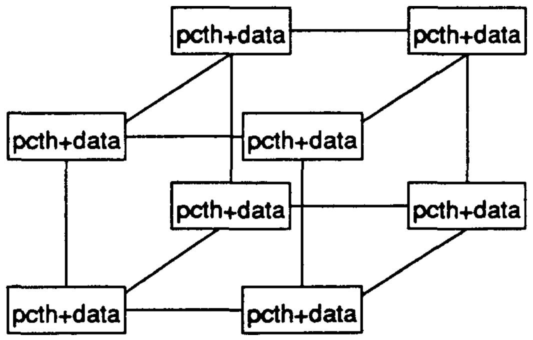

FIGURE 2
Generic MIMD configurations for PCTH.

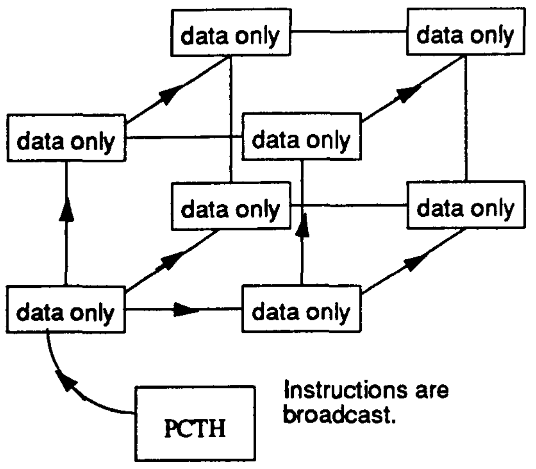

FIGURE 3
Generic SIMD configurations for PCTH.

code actually runs on the front end machine and broadcasts instructions for operations to be taken on parallel data objects. A conceptual view of the SIMD architecture for PCTH is shown in Figure 3

The CRAY Y-MP at Sandia has 8 CPU's rated at about 340 MFLOPS each with vector chaining. Total memory amounts to 64 Mwords with a 256 Mword Solid State Disk. A short comparison will reveal that these machines are roughly comparable in speed and memory capacity.

3. PCTH DEVELOPMENT

The development of the SIMD and MIMD versions of PCTH has proceeded at about the same pace. To date we have ported only the computational kernel of CTH for 2 dimensions while leaving the pre- and post- processing software alone (Figure 4). Code running on the host is used to obtain the database from the front end machine (Sun 4/490) and partition the data to the nodes. The earliest MIMD work was done on a hypercube simulator running on a workstation. The problem domain is subdivided into overlapping subregions. Each node (operating on it's own subregion) proceeds with the algorithm until boundary information is required from neighboring processors. Once the boundary information is received then each processor proceeds again until the next synchronization point and so on. Significant portions of the rezone step needed to be re-written in order to provide a simple interface for updating boundary variables and also to provide a type of conceptual portability to the CM code. The nCUBE and CM codes are written in Fortran 77 and Fortran 90, respectively. The major required feature in Fortran 90 is the array construct which allows one to reference a large parallel variable or array as a single entity. To a great extent the idea of array objects (although not the syntax) was maintained in the MIMD code in order to be able to easily compare with the SIMD code (conceptual portability). In fact this concept was an essential element in developing a fairly simple programming paradigm for internode communications. The original compatible versions of the SIMD and MIMD codes included only an ideal gas equations of state for plane two- dimensional modeling but did include the basic mass, momentum and energy conservation along with multiple material interface tracking. We did not include the options

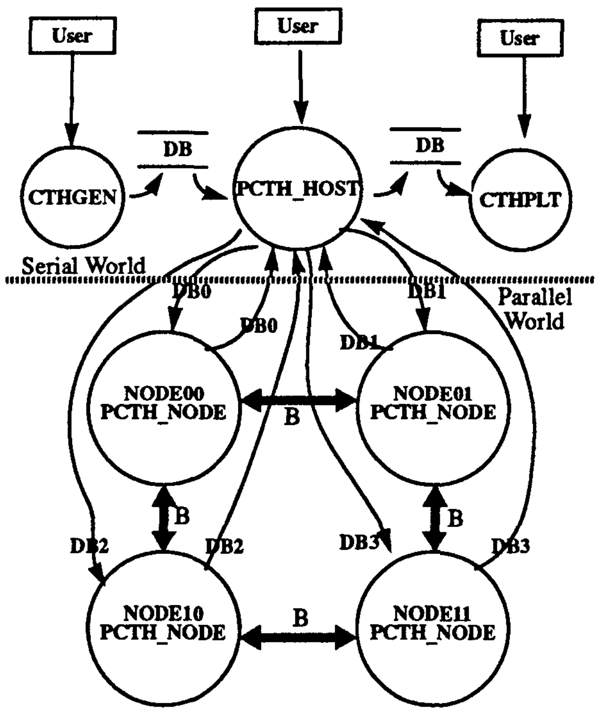

FIGURE 4
Dataflow diagram for the MIMD version of PCTH.

	Mesh Size		
Machine	128^2	256^2	512^2
PCTH, CM-2, 16K, slicewise	11.5	7.1	5.6
PCTH, nCUBE2, 1024 nodes	7.9	4.1	3.1
CTH, CRAY Y-MP, 1 CPU	20.0	16.0	15.0

TABLE 1
Grind time (cpu time in μs per cell per cycle.[4])

in the CTH production code which require mixed cells to have common pressures and/or temperatures as this could lead to serious load imbalance due to the iterations which are required. Instead, only the most commonly used mixed material cell model in CTH (MMP)[3], which computes pressures and temperatures separately for each material in each cell, was implemented since it is highly parallel. The code was used to run benchmark calculations to show proof of concept. The results of these calculations are shown in Table 1. The table shows that in terms of speed the parallel architectures are quite competitive with a current generation vector supercomputer. Further optimization of the massively parallel codes would improve the performance even more. Note that the performance of the massively-parallel computers improves as the problem size increases. This is due to the fact that the communication overhead begins to become insignificant relative to computation time as the amount of the computational work assigned to each processor increases.

The PCTH codes are in a state of rapid development. At the date of this writing, mass, momentum and energy conservation in 2D plane and cylindrical geometries, multiple material interface tracking, periodic and rigid boundary conditions, ideal and Mie-Gruneisen equations of state, programmed burn high explosive modeling with JWL EOS, von-Mises yield stress elastic-plastic material modeling and fracture based on a minimum pressure criterion have been implemented. We are beginning to characterize the performance of the codes on initial prototype production modeling problems.

4. POSTPROCESSING

Both the CM and the nCUBE have graphics framebuffers which we utilize extensively for debugging and demonstration. Watching a graphical display of state variables as the simulation progresses conveys both the dynamics of material interactions, and any computational anomalies. We also write data to parallel disk arrays, and display the image sequence (movie) after the calculation is complete.

5. FUTURE PLANS

We intend to proceed with the development of the PCTH codes, both to advance simulation modeling capability and to provide an optimized and productive tool. We will characterize load imbalance issues, especially those introduced by sophisticated material response modeling, and develop techniques for reducing these effects. We will also develop the capability for analysts at remote workstations to interact with databases created and stored in the parallel world. This will remove the serial bottleneck at the host program illustrated in Figure 4 and provide the capability for more analysts to use the system.

ACKNOWLEDGEMENTS

We would like to thank Mike McGlaun for his constant encouragement and enthusiasm for this development work.

REFERENCES

1. J. M. McGlaun, S. L. Thompson, and M. G. Elrick, CTH: A Three-Dimensional Shock Wave Physics Code, Int. J. Impact Engng. 10 (1990) 351.
2. A. C. Robinson, E. Fang, D. Holdridge, J. M. McGlaun, A Development Plan for a Massively-parallel Version of the Hydrocode CTH, Sandia Report, SAND90-0589, 1990.
3. J. M. McGlaun, CTH Reference Manual: Cell Thermodynamics, Sandia Report, SAND91-0002, 1991.
4. H. E. Fang, A. C. Robinson and Kah-Song Cho, Hydrocode Development on the Connection Machine, Proc. 5th SIAM Conference on Parallel Processing, 1991.