Scalable computations in penetration mechanics

K. D. Kimsey*, S. J. Schraml* & E. S. Hertel†

* *U.S. Army Research Laboratory, Weapons and Materials Research Directorate, Aberdeen Proving Ground, MD 21005-5066*

† *Sandia National Laboratories, Albuquerque, NM 87185*

This paper presents an overview of an explicit message-passing paradigm for an Eulerian finite volume method for modeling solid dynamics problems involving shock wave propagation, multiple materials, and large deformations. Three-dimensional simulations of high-velocity impact were conducted on the IBM SP2, the SGI Power Challenge Array, and the SGI Origin 2000. The scalability of the message-passing code on distributed-memory and symmetric multiprocessor architectures is presented and compared to the ideal linear performance.

1 INTRODUCTION

The mechanics of penetration and perforation of solids has long been of interest for military applications in terminal ballistics. Kinetic energy penetration phenomena are also germane to applications involving high-mass and high-velocity debris due to accidents or high-rate energy release, the transportation safety of hazardous materials, the safety of nuclear reactor containment vessels, the design of lightweight body armors, the erosion and fracture of solids due to repeated impacts by liquid or solid particles, and the protection of spacecraft from meteoroid impact. A thorough review of the fundamentals of penetration and perforation and their application to practical problems has been prepared by Goldsmith,[1] Johnson,[2] Backman and Goldsmith,[3] and Zukas et al.[4,5]

Analytical approaches to penetration mechanics tend to fall into three categories: empirical or quasi-analytical, approximate analytical, and numerical methods. While empirical and approximate analytical methods are quite useful for developing an appreciation for the dominant physical phenomena, they are limited in scope. Numerical methods provide a complete description of the dynamics of impacting solids accounting for the geometry of the interacting bodies; elastic, plastic, and shock wave propagation; hydrodynamic flow; finite strains and deformations; high strain rate material behavior; and the initiation and propagation of failure in the colliding bodies. Computer codes for modeling wave propagation and impact have matured considerably since their initial development some 45 years ago. Today they serve as valuable tools in studies of materials and structures subjected to intense impulsive loading. Recently, Benson[6] documented a comprehensive review of the physics and numerics in wave propagation codes.

Three-dimensional simulations of high-velocity impact phenomena continue to delineate the high performance computing resources for Army applications in terminal ballistics. Current applications in high-velocity impact phenomena require that the simulation time increase from the microsecond to millisecond regime; complex geometries dictate a finer mesh resolution that mandates a smaller time integration increment to

satisfy stability criteria and additional time integration cycles. Memory requirements for large-scale Eulerian finite volume simulations scale with the cube of the zone size (i.e., for a fixed computational domain). Doubling the number of zones in each coordinate direction by halving the characteristic zone length increases the memory requirement by a factor of 8 and halves the time step. These factors, when coupled with the requirement to model larger physical domains, are strong stimuli for exploiting scalable architectures and algorithms.

Under the aegis of the DOD High Performance Computing (HPC) Modernization Program,[7] DOD researchers are afforded access to scalable HPC computing resources. The successful utilization of scalable architectures for large-scale simulations of high-velocity impact requires reliable and robust scalable applications algorithms. The Common HPC Scalable Software Initiative (CHSSI) component of the DOD HPC modernization program addresses the development, validation, and demonstration of scalable software in a number of defense computational technology areas. This paper presents an overview of an explicit message-passing paradigm for applications in shock physics. Scalable performance of a three-dimensional oblique rod impact is presented for distributed-memory and symmetric multiple processor architectures.

2 SCALABLE PARADIGM FOR IMPACT PROBLEMS

CTH[8] is an Eulerian finite volume code for modeling solid dynamics problems involving shock wave propagation, multiple materials, and large deformations in one, two, and three dimensions. CTH is widely used across the defense research and development community to model problems in shock wave propagation. CTH employs a two-step solution scheme; a Lagrangian step followed by a remap step. The conservation equations are replaced by explicit finite volume equations that are solved in the Lagrangian step. The remap step uses operator-splitting techniques to replace multidimensional equations with a set of one-dimensional equations. The remap or advection step is based on a second-order-accurate van Leer[9] scheme. High-resolution material interface trackers are available to minimize material dispersion. Both analytical and tabular equations of state are available to model the hydrodynamic behavior of materials. Models for elastic-plastic behavior and high-explosive detonation are also available.

Robinson[10] developed the algorithmic framework for conducting scalable Eulerian finite volume simulations for modeling problems in solid dynamics based on object-oriented programming. Robinson demonstrated that the structured mesh of the Eulerian finite volume method is well suited for scalable paradigms employing message passing between computational subdomains.

Distributed-memory, scalable architectures are characterized by a large number of discrete computational nodes consisting of memory, a commodity CPU chip, and access to an internal communications network. One computing technique that can be employed on this architecture is referred to as single program multiple data (SPMD). The SPMD technique has the same executable running on each computational node, but each executable is working on a different data set. Algorithms that depend on a fixed logically connected mesh are relatively simple to map onto an SPMD machine. The technique used for CTH is similar to that developed by Robinson,[10] in that the entire problem domain is broken up into subdomains that reside on individual computational nodes. Communication between subdomains is handled by the use of "ghost" cells and explicit messages that are passed between nodes. The use of "ghost" cells is a common technique for applying boundary conditions (i.e., finite difference scheme) independent of edges and corners in Eulerian codes. At an external boundary, the ghost cell data are based on the selected boundary condition approximation. Similarly, at a subdomain or internal boundary, the ghost cells contain data acquired in a message passed from a neighboring node. A simple example of mesh decomposition is displayed in Fig. 1.

CTH is a set of codes that work together to solve the conservation equations of mass, momentum, and energy for shock physics simulations. The two

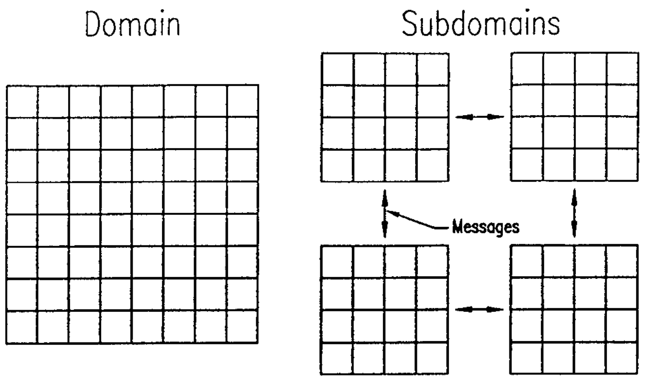

Fig. 1. CTH explicit message-passing paradigm.

primary codes for modeling solid dynamics problems are CTHGEN and CTH. CTHGEN reads the user input and builds a time-zero representation of the problem specifications. CTH reads additional user input and the time-zero data from CTHGEN (through a disk file known as a "restart" file) and initiates the time integration process. A similar computational paradigm for problem generation or initialization and integration of the governing equations is employed on scalable architectures. A copy of CTHGEN runs on node 0, reads the user input, and broadcasts the input to all other nodes. Based on the total number of nodes requested by the user and the total problem size, CTHGEN performs the mesh decomposition to map the problem space onto the nodes. The mesh decomposition algorithm builds near-cubic computational subdomains that equalize the amount of work on each node (equal number of cells per node). The requirement for cubic subdomains is for two reasons: (1) to minimize the surface-to-volume ratio, and (2) to keep the size of explicit messages between nodes as equal as possible. Once the problem is mapped onto the available nodes, the individual copies of CTHGEN insert material into the respective subdomains, define material properties, and complete the time-zero representation of the numerical model.

Time integration of the governing equations is controlled by CTH. A copy of CTH runs on node 0, reads the user input, and broadcasts the input to all other nodes. Once the broadcast is complete, each node reads its individual restart or database file. At this point, the time integration starts. The solution sequence for CTH has not changed for scalable architectures; a Lagrangian step followed by a remap step. Essentially, every time ghost cell values change, CTH exchanges these new values with neighboring nodes before the updated values are used in the solution sequence. The overall solution sequence of CTH is a Lagrangian step followed by a remap and then a database modification step where materials may be discarded or velocity transformations applied. The Lagrangian step consists of several tasks. The first task defines the artificial viscosity. During this task, messages are exchanged with neighbors to calculate the correct boundary values. Once the artificial viscosity and pressure are defined, CTH calculates new cell velocities that are exchanged with neighboring nodes. The new cell velocities are used to update the stress deviators at this time, and then the new stress deviators are exchanged with neighboring nodes. The new stress deviator information leads to new energy terms from the updated work terms. After the energy is updated, the Lagrangian step is effectively complete. Special models like the multiphase reactive flow package perform tasks to prepare for the remap step. At the end of the Lagrangian step, all ghost cell values are exchanged for the last (fourth) time.

CTH uses a second-order-accurate advection scheme. This scheme, based on work by van Leer,[9] determines a linear slope across each "donor" cell. To calculate this slope, data from three cells are required: the donor cell, the cell upstream, and the cell downstream. At an internal or subdomain boundary, a given node "knows" the values in the ghost cell, the first real cell, and the next adjacent real cell near the internal boundary. If the flow is "outward" (from the last real cell into the ghost cell), the code has enough information to calculate the slope across the donor cell (the last real cell). However, if flow is from the ghost cell into the first real cell, the code does not have information about the "upstream" cell (the cell beyond the ghost cell). But, if a node shares this boundary, the inflow for this node is exactly the same as the "outflow" for the adjacent node. A new set of subroutines that calculate outflow values for each node boundary has been developed. These values are collected and passed to the adjacent nodes. If a node calculates an "inflow" value and finds that another node shares that boundary, the value from the message is used rather than the incorrectly calculated value. By using the second-order-accurate outflow value from the adjacent node, results from single-node simulations can be exactly duplicated on muliple-node simulations.

Several exchanges must be completed during the Eulerian (or remap) step. As noted previously, each time ghost cell values are modified, data must be exchanged with neighboring nodes. The first task in the remap step converts cell volume fractions to volumes, which requires new ghost cell data to be exchanged with neighbor nodes. CTH uses an operator-splitting technique for the remap step. Each time the remap is completed in a particular coordinate direction, the ghost cell values are exchanged with neighboring nodes. During this step, momenta and updated mass values are also exchanged because CTH uses the half index shifted momentum advection scheme of Benson.[11] This method requires correct velocities at the node mesh boundaries. Since velocities on the edges of isolated material cells are also modified during the remap step, these corrected velocities must also be exchanged. Finally, after all remap steps have been completed, the volumes are converted back to volume fractions and the Eulerian energy balance is accomplished. This step calls the equation of state for each material yielding new cell pressures, temperatures, and sound speeds. One of the last steps in the remap is to calculate the minimum time step. This is first done for each subdomain, and then a global minimum over all subdomains is calculated to determine the time step. This is the last time that significantly sized messages are exchanged.

An additional feature that needs to be addressed for scalable architectures is the use of tracer particles or data collection points. CTH records flow field data at tracer particle locations as the simulation progresses. The particles can either move with the bulk flow field or be fixed in space. Each tracer particle is initially placed in the mesh based on user specified coordinates. If the tracer particle coordinates are in the interior of the volume of a computational subdomain, the coordinates are recorded by that computational node and a flag is set in the tracer data storage. This tracer particles' coordinates in all other subdomains are recorded as (1.0e20, 1.0e20, 1.0e20), the upper right-hand coordinate of the universe. In addition, the tracer particle location flag is set to indicate nonownership by that node. This flagging technique permits a quick check on whether or not a particular tracer is active in a given subdomain or node. At the end of each time step, the tracer particles' coordinates are updated by the nodes "owning" the tracers. Messages are then exchanged between nearest neighbor nodes. After all six messages have been exchanged, all 27 nodes surrounding the actual position of the tracer particle know the true coordinates. These coordinates are then compared with the limiting coordinates for each node. If the tracer has migrated to a new node, the new coordinates are set in the tracer array and the tracer flag is reset to indicate ownership. The tracer coordinates for all other nodes are set to (1.0e20, 1.0e20, 1.0e20). There is no need to propagate the coordinates to nodes beyond nearest neighbors since the time step controls prevent any tracer particle from moving more than one cell width in any given time step.

For a three-dimensional calculation, approximately 24 large messages are passed during the Lagrangian step, and approximately 48 large messages are passed during the Eulerian step. A large message contains all cell variables on the face adjoining two nodes. Typical problems consist of 40–80 variables per cell. Therefore, large messages are at least 200–400 kB. Several small messages are exchanged during the solution sequence. These messages are typically exchanged during the calculation of global sums and minimization processes, such as calculation of the global time step. All message passing is done through an interface independent code specific software layer. This technique allows support of both serial and parallel code versions in a simple, yet effective fashion. Scalable performance discussed in this paper is based on message-passing-interface (MPI)[12,13] routines for explicit message passing.

3 SCALABLE ARCHITECTURES

The IBM SP2 located at the Aeronautical Systems Center (ASC),Wright-Patterson Air Force Base, was used to conduct scalability studies on the distributed-memory architecture. This system is comprised of 256 R6000 processor elements, 233 of which are used as dedicated compute nodes. Each compute node has 1 GB of memory.

Scalability studies were also conducted on two different types of Symmetric Multiple Processor (SMP) systems, the SGI Power Challenge Array (PCA) and the SGI Origin 2000 (O2K). The PCA system that was employed for the study is located at the U.S. Army Tank-Automotive Research, Development, and Engineering Center (TARDEC). This system consists of four nodes, each containing 16 R10000 processors, 4 GB of memory, and a high performance parallel interface (HIPPI). For parallel CTH calculations utilizing processors on multiple nodes, TCP sockets were used to transfer messages across the HIPPI channels for internode communication.

Two different O2K systems were also used in the study, one of which is located at the U.S. Army Research Laboratory (ARL) and the other at ASC. The ARL system consists of 32 R10000 processors and 12 GB of memory. At the time this study was performed, the ASC system consisted of seven individual O2K systems, each with 32 R10000 processors and 16 GB of memory. Limitations of the batch-queuing software on the ASC system prohibited running parallel calculations across multiple O2K hosts. Thus, only one of the seven ASC O2K systems was used in the scalability study.

4 SCALABLE HIGH-VELOCITY IMPACT SIMULATIONS

CTH with explicit message passing has been used to model a long-rod projectile impacting an oblique steel plate on both distributed-memory and symmetric multiple processor architectures. This problem was selected due to well-characterized experimental data reported by Fugelso and Taylor[14] and previous serial CTH simulations conducted by Hertel.[15] Fugelso and Taylor conducted a series of ballistic experiments to evaluate the effects of combined obliquity and yaw on high-density long-rod projectiles. Depleted uranium alloy long-rod projectiles with no yaw were obliquely launched into a rolled homogeneous armor (RHA) plate that had been accelerated by an explosive charge, resulting in a yawed impact in the plate frame of reference. The depleted uranium alloy (DU 0.75%Ti) projectiles were right-circular cylinders with a hemispherical nose, and the impact velocities ranged from 0.85–1.65 km/s. Yaw and obliquity angles ranged from 0–70° and 10–0°, respectively, in the test series. The length and diameter of the projectile in shot 58 of the test series are 7.67 cm and 0.767 cm, respectively, for a length-to-diameter (L/D) ratio of 10. The striking velocity was 1.289 km/s, and the thickness of the RHA was 6.4 mm. In the laboratory frame of reference, the angle of obliquity was 73.5°, the plate velocity was 0.217 km/s, and the projectile velocity was 1.21 km/s. In the plate frame of reference, the angle of obliquity was 64.2°, the projectile velocity was 1.289 km/s, and the yaw angle was −9.3°. A schematic of the initial condition for shot 58 is illustrated in Fig. 2.

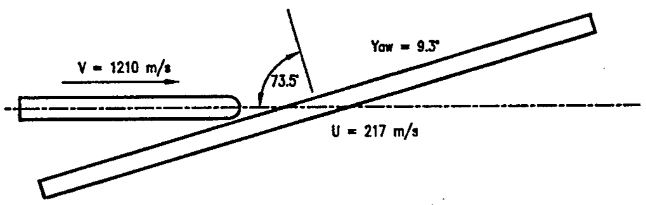

Fig. 2. Initial conditions for combined yaw and obliquity simulation.

The scalability study was carried out maintaining a constant work load (i.e., the number of computational cells on each processor for each of the simulations). This was done to keep the computation-to-communication ratio constant for simulations involving different numbers of processors. Maintaining a constant computation-to-communication ratio and eliminating disk access for intermediate plot and restart files during the time integration permitted the computational performance to be isolated and measured as a function of the number of processors used.

The single processor, baseline calculation used a Cartesian computational domain spanning 21.5 cm in the X direction, 3.0 cm in the Y direction, and 6.0 cm in the Z direction. The computational domain was discretized into uniform cubic zones of length 0.1 cm, resulting in a three-dimensional grid of 215 × 30 × 60. The scalability study used power-of-two sets of processors, and the number of zones in the model was varied to maintain a nearly constant number of computational zones per processor.

All calculations were conducted for a simulated time of 40 μs. The grid was incrementally refined by uniformly decreasing the characteristic zone length in each coordinate direction by a factor of $2^{-1/3}$. This approach approximately doubles the number of grid points with each successive mesh refinement. The characteristics of the grids used in the scalability study are summarized in Table 1. The number of zones (NI, NJ, and NK) in each coordination direction (X, Y, and Z) listed in Table 1 does not include ghost cells. An alternative to this mesh refinement technique would have been to double the number of zones in one direction for one refinement, then double the number of zones in another direction for the next refinement, and so on. This approach would reduce the time step by a factor of 2 on the first refinement and would have doubled the number of time integration cycles (i.e., computational cycles) to reach the desired simulation time of 40 μs, whereas the method of uniform zone size reduction reduces the time step by a factor of $2^{-1/3}$ with each refinement. As a result, the number of computational cycles required to reach 40 μs of simulated time increased only by a factor of approximately $2^{1/3}$ as the number of processors was increased.

The scalable performance of the message-passing code is measured by the "grind time," which is the average CPU time required for the code to update all flow field variables for one computational cell in a given time increment (cycle). The grind time is expressed in units of μs/zone-cycle. The grind time for ideal linear scalability decreases by a factor of 2 for every doubling of processors used if the work per node is constant.

The results of the CTH scalability study on the IBM SP2 are presented graphically in Fig. 3. A maximum of 64 processors was used on the SP2. The table illustrates the effect of decreasing zone size on the number of integration cycles required to reach the simulation time of 40 μs. With each successive refinement, the reduction of the time step results in an increase in the required number of cycles by a factor of approximately $2^{1/3}$.

Table 1. Computational grids used in scalability study

No. of Proc.	NI	NJ	NK	Total No. of Zones	Ave. No. of Zones/Proc.	Char. Zone Length (cm)
1	215	30	60	387,000	387,000	0.100
2	271	38	75	772,350	386,175	0.080
4	341	48	95	1,554,960	388,740	0.063
8	430	60	120	3,096,000	387,000	0.050
16	541	76	151	6,208,516	388,032	0.040
32	683	95	191	12,393,035	387,282	0.031
64	860	120	240	24,768,000	387,000	0.025

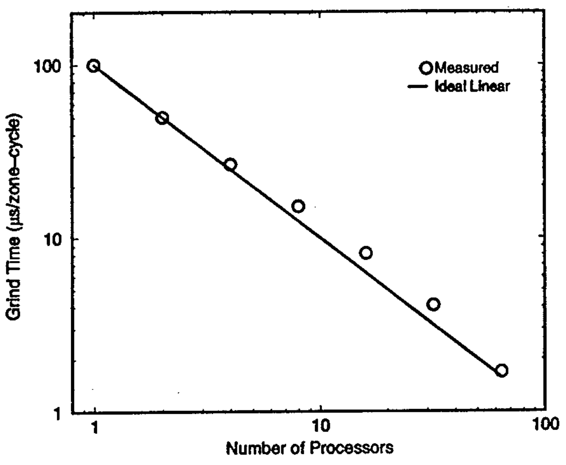

Fig. 3. Measured vs. Linear scalability performance on IBM SP2.

Figure 3 shows the measured grind time results as a function of the number of processors used. These data are compared to the "ideal linear" performance that is obtained by dividing the single processor grind time by the number of processors used. When plotted on a log-log scale, the ideal linear performance data form a straight line. This figure shows that the performance of message-passing CTH on the SP2 does scale with the number of processors used. The results of the intermediate processor sets (8, 16, and 32 processors) fall slightly off of the ideal performance line. However, the measured results return to the ideal performance line for the 64-processor result.

Tests on the SGI Origin 2000 systems located at ARL and ASC were limited to a maximum of 16 processors. As described earlier, each system has a total of 32 processors. All of the scalability tests were performed using power-of-two sets of processors. In order to avoid contention between the operating system and the parallel application, a decision was made to avoid running calculations in which all of the processors were used by the application.

Figure 4 is a plot of the measured grind times as a function of the number of processors. The results of the ARL system are represented by the circle symbols, and results from the ASC system are represented by the squares. There are also two straight lines on this plot, one of which is labeled "Ideal (m = 1.000)" and the other is denoted as "Linear (m = 0.727)." The "ideal" line extends below the measured data and the "linear" curve. Like the straight line in Fig. 3, this line represents the ideal scalability of the application on the parallel computer as represented in eqn (1):

$$\text{grind time}_n = \text{grind time}_1/n \tag{1}$$

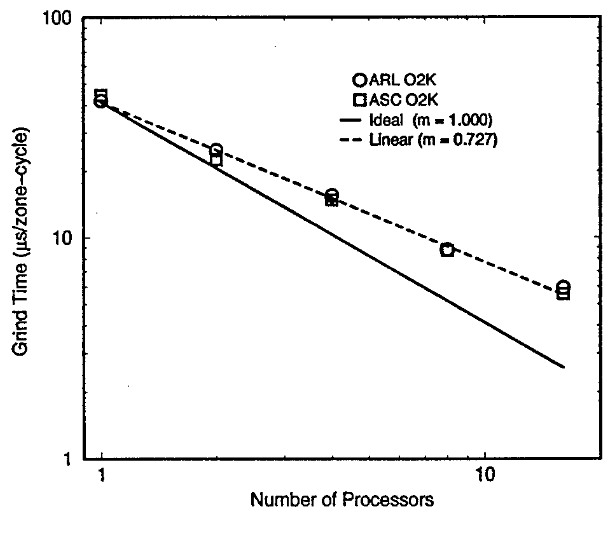

Fig. 4. Measured vs. linear scalability on SGI Origin 2000.

The measured data do not follow this ideal scalability line, but do appear to form a straight line on the log-log plot. By computing the log of the measured data (number of processors and grind times), linear regression may be used to determine the slope of the line that best fits the measured data. The expression that represents this best fit line is presented in eqn (2):

$$\text{grind time}_n = 10^b/n^m, \tag{2}$$

where the values of m and b are the slope and intercept of the line from the regression analysis. The value of m can be considered to represent the parallel efficiency of the application on that particular system. The value 10^b closely approximates the measured grind time on a single processor. For the ideal scalability case, substitution of m = 1.0 into eqn (2) yields eqn (1). The fact that near ideal linear performance was obtained on the IBM SP2 proves that the message-passing CTH algorithm is not the source of the less-than-optimum parallel efficiency of the O2K. These results suggest that either the message traffic between processors or the movement of data from main memory to the individual processors is the limiting factor in the scalability of the O2K system. Tests on large parallel systems show that nearly ideal scalability extends to greater than 2,000 processors.

The SMP architecture scalability study was performed on the SGI PCA using power-of-two sets of R10000 processors. As stated earlier, this system consists of four nodes, each with 16 processors. In a manner similar to the O2K tests, tests were avoided in which all processors on a node would be used. Thus, the scalability tests involved a maximum of eight processors per node. The results of the tests are summarized and plotted in Fig. 5. The four-node calculations using 16 and 32 processors were not completed to 40 μs at the time this paper was written. Grind times from simulations that were run to 1 μs were used as a substitute.

In the plot in Fig. 5, the grind time results from the single-node calculations are represented by the circle symbols, squares are used to represent the

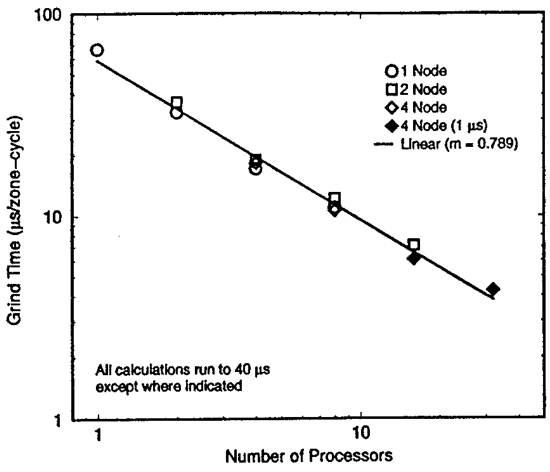

Fig. 5. Measured performance on SGI PCA at TARDEC (R10000 processors).

two-node results, and diamonds identify the four-node results. The four-node results on four and eight processors that were run to 40 µs use an empty diamond, and the filled diamonds represent the 16- and 32-processor calculations that were run to 1 µs. This figure shows that the single-node and multinode results form a straight line on the log-log plot signifying linear scalability. The agreement between the single- and multiple-node results indicates that the communication overhead for messages passed between nodes is not significantly greater than that of messages passed between processors of a given node. Like the O2K results, the measured data were used to perform a regression analysis, the results of which are represented by the straight line in the figure. This regression analysis produced a parallel efficiency value, m, of 0.789.

As described earlier, scalability tests on the O2K and PCA systems were limited to a maximum of half the number of available processors per system (16 processors on the 32-processor O2K, and 8 processors on the 16-processor PCA nodes). This was done to avoid possible contention between the application software and the operating system in the event that all of the available processors are requested for the parallel CTH calculation. To determine the impact of such a situation on the parallel performance of the code, a series of tests was performed on a 16-processor node of the TARDEC PCA. Each calculation was run independently for a simulation time of 1 µs, with no other applications running. Three sets of problems were run using 1–16 processors. One of the problem sets used identical problem definitions from the 40-µs simulations and contained approximately 387,000 computational zones per processor (not including ghost cells). The other two sets used half and double the number of computational zones at the first set (193,500 zones per processor and 774,000 zones per processor). The measured results from these three sets of calculations are presented in Fig. 6. Figure 6 illustrates the grind time as a function of the number of processors for the three sets of calculations. The results for 1–8-processors fall in an approximately straight line. The data from the 1–8 processor results were used in a regression analysis to generate the straight line in the figure. The tests that used 10 or more processors (more than half the total number of processors in the machine) produced grind times that gradually grew away from the regression line and either reached an asymptotic value or began to increase, depending on the problem size.

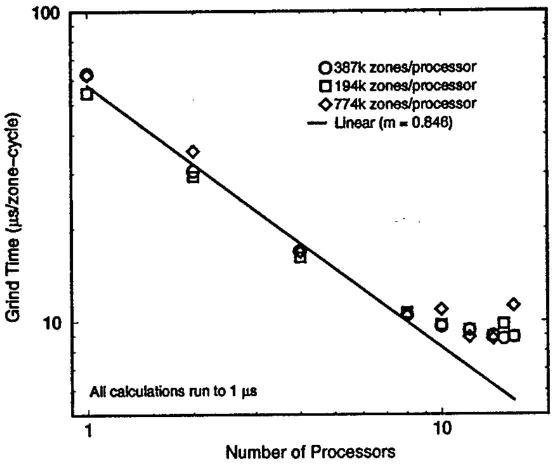

Fig. 6. Single-node measured performance for 1 µs on SGI PCA (R10000 processors).

5 CONCLUSIONS

Three-dimensional simulations of a long-rod projectile impacting an oblique steel plate were conducted on the IBM SP2, SGI PCA, and the SGI O2000 systems as part of a scalability study. The scalability study was conducted using a constant work load (i.e., number of computational cells on each processor). This approach maintained a constant computation-to-communication ratio and employed an average of 387,000 cells per processor. A maximum of 64 processors (~25 million computational cells) was used in the scalability study.

Scalable performance of an explicit message-passing implementation of CTH was measured by the "grind time," which is the average CPU time required for the code to update all flow field variables for one computational cell in a given time increment (cycle). Scalable performance on the distributed memory IBM SP2 system demonstrated near "ideal" linear performance based on measured grind times for 1–64 processors. The measured grind times on the SGI O2000 and SGI PCA do not follow the ideal linear scalability, but do appear to form a straight line on a log-log plot. In addition, scalable performance on the SGI PCA (R10000 processors) using multiple nodes indicates that the communication overhead for messages passed between nodes is not significantly greater than that for messages passed between processors on a given node.

REFERENCES

1. Goldsmith, W., *Impact,* Edward Arnold, London, 1960.
2. Johnson, W., *Impact Strength of Materials*, Crane, Russak, New York, 1972.
3. Backman, M. E., and Goldsmith, W., "The mechanics of penetration of projectiles into targets," *Int. J. Engrg. Sci.,* 1978, **16**, 1–99.
4. Zukas, J. A. et al., *Impact Dynamics*, Wiley-Interscience, New York, 1982.
5. Zukas, J. A. et al., *High Velocity Impact Dynamics*, Wiley-Interscience, New York, 1990.
6. Benson, D. J., "Computational methods in Lagrangian and Eulerian hydrocodes," *Comput. Methods Appl. Mech. Engrg.*, 1992, **99**, 235–394.
7. Jones, A. K., "Modernizing high performance computing for the military," *Computational Science and Engineering,* 1996, **3**(3), 71–74.
8. McGlaun, J. M., and Thompson, S. L., "CTH: A three-dimensional shock wave physics code," *Int. J. Impact Engng.*, 1990, **10**, 351–360.
9. Van Leer, B., "Towards the ultimate conservative difference scheme IV, a new approach to numerical convection," *J. Comp. Phys.*, 1977, **23**(276).
10. Robinson, A. C., Ames, A. L., Fang, H. E., Pavlakos, C., Vaughan, C. T., and Campbell, P., "Massively parallel computing, C++ and hydrocode algorithms," Proceedings of the Eighth Conference in Computing in Civil Engineering, Dallas, TX, 1992.
11. Benson, D. J., "Momentum advection on a staggered mesh," *J. Comp. Phys.,* 1991, **100**(1).
12. Gropp, W., Lusk, E., and Skjellum, A., *Using MPI,* MIT Press, ISBN 0-262-57104-8, 1994.
13. Snir, M., Otto, S. W., Huss-Lederman, S., Walker, D., and Dongarra, J., "MPI: The Complete Reference," MIT Press, ISBN 0-262-69194-1, November 1995.
14. Fugelso, E., and Taylor, J. W., "Evaluation of combined obliquity and yaw for U-0.75% Ti penetrators," LA-7402-MS, Los Alamos National Laboratory, Los Alamos, NM, 1978.
15. Hertel, E. S., "A comparison of the CTH hydrodynamics code with experimental data," SAND92-1879, Sandia National Laboratories, Albuquerque, NM, 1992.

M98002636

Report Number (14)	SAND--98-0254C
	CONF-9710135--
Publ. Date (11)	199801
Sponsor Code (18)	DOE/DP, XF
UC Category (19)	UC-700, DOE/ER

DOE