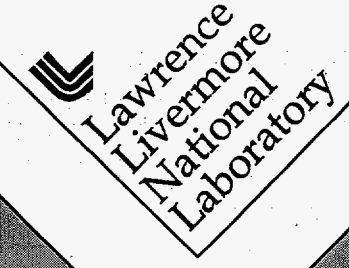
CONF-9604162--1
UCRL-JC-123339
PREPRINT

Status of ParaDyn: DYNA3D for Parallel Computing

G. L. Goudreau
C. G. Hoover
A. J. DeGroot
P. J. Raboin

This paper was prepared for submittal to the
Workshop on Recent Advances in Computational Structural Dynamics
and High Performance Computing
Vicksburg, Mississippi
April 24-26, 1996

April 17, 1996



Status of ParaDyn: DYNA3D for Parallel Computing

G. L. Goudreau, C. G. Hoover, A. J. De Groot, and P. J. Raboin,
Lawrence Livermore National Laboratory

The evolution of DYNA3D from a vector supercomputer code into a parallel code is reviewed. Current status and target applications are suggested, especially those of interest to the Department of Defense.

It was no coincidence that the development and success of the Livermore nonlinear solid and structural mechanics codes DYNA (Goudreau and Hallquist, 1982, Whirley, 1993) and NIKE (Maker et al, 1991) occurred during the rise and preeminence of the vector supercomputers epitomized by Cray Research, from the first Cray-1 installed here in 1976 through the current C-90 megalith. The tremendous vector programming efficiencies achieved in a single processor by element chunking, (Goudreau, 1982) as well as optimization of direct linear equation solvers in NIKE, have masked the fact that for most of the last fifteen years those Cray machines have been multi-pipe (or parallel) machines with large (fast but expensive) shared memory. However, in a production environment, with several hundred users demanding equal access, and often with multiple jobs to be executed per user, computer system managers have found it optimal to discourage individual codes from using multiple pipes for parallel applications, and have instead optimized the system parameters to timeshare the multiple single processor jobs. The National Energy Research Supercomputer Center (NERSC) has had more of a computing research environment, and multitasking of Cray applications has been available for moderate numbers of processors from eight to sixteen. These days much more cost-effective, high-end workstations coming from other vendors are attempting this programming class for one to two orders of magnitude lower cost.

The heavy cost of large shared memory, at least as much as diminishing returns on development of expensive single processor pipes, has led the industry (even Cray) into distributed memory massively parallel machines. Intel, with its Paragon seems the only major vendor headed toward thousands of processors. The failure of Thinking Machines Corporation and its CM-5, and even the corporate difficulties of Meiko and its CS-2 investment here at Livermore show the pitfalls of this very high end, primarily capability based machine market. Cray, after its recent acquisition by Silicon Graphics Incorporated, is probably strong enough to cover its investment in the T3D (and presumably the T3E to come). The last five years have seen a moderate investment in parallelism in DYNA (Hoover et al, 1994, 1995). The ParaDyn project was launched by joint funding by institutional R&D and the weapons engineering program. This has required close cooperation between the small team developing the new DYNA prototype, without getting too out of touch

with the main production serial-vector team. Repository control within each team, with periodic merging of the two repositories, has been essential.

Our first experiment was the complete conversion of DYNA2D to parallel FORTRAN (CM90) on the CM-5 at the Army's High Performance Computing Center at the University of Minnesota. While an interesting experience in the data parallel programming model, the serious departure from the norm of engineering computing software, the poor success of Thinking Machines in making this machine work for finite element (or unstructured grid) applications, and the lack of sufficient vendors seriously pursuing the data parallel programming model led to our decision to not attempt a data parallel version of DYNA3D. HPF had not yet arrived, and the alternative message passing strategy is more heavily supported by others, both hardware vendors and applications programs.

Four years into this paradigm shift, the ParaDyn team reports that both DYNA3D and NIKE3D (with their basic vector organization), are well suited to the modular insertion of message passing protocols to communicate between processors, leaving the vendors to compete on optimal implementation of common standards (e.g. PVM and MPI). While much remains to be done on the mechanics side of code organization, especially on the implicit NIKE side, more serious challenges remain on the infrastructure issues of extending mesh generation beyond the million zone problem, including the domain decomposition algorithms, the parallel composition of graphic output, and long term data storage and retrieval.

Current massively parallel computers consist of a set of processors connected with a high speed network. The DYNA3D parallel algorithms are designed to divide the problem mesh into sub domains and to execute the full DYNA3D program on a sub domain in each processor. Solutions on sub domains are connected together by communicating data across processors along the boundaries of the sub domains. Algorithms assign elements (continuum, shell, and beam) to one processor uniquely, and nodes to any processor owning a connected element. Consequently, nodes on processor boundaries, shared nodes, are stored in more than one processor. Boundary node lists are grouped by target processor so only one send per processor time step need be executed for each other processor with which a given processor interacts. The decoupled time integration for the nodal coordinates is duplicated for the shared nodes. The motion for shared nodes is duplicated identically by adding a communication step after the nodal forces are calculated. For example, the force on a shared node due to internal deformations of the elements is a sum of the partial force calculated from connected elements in the same processor and connected elements in other processors. In the more general case, the total nodal force is a sum of contributions from applied loads, element deformations, contact-restoring forces, and other boundary conditions. Duplicate nodal motion calculations also requires

communicating partial mass accumulation corresponding to partial force accumulation. But in a Lagrangian code, mass is conserved, and a partial-mass communication step occurs only at initialization, or after rezoning. The additive force contributions allow separate parallel treatment for the element deformation mechanics, contact algorithms, and boundary forces.

The finite element method has long popularized the ideal of a global-local hierarchy. One is used to localization of global quantities (e.g. motions) to an element, processing an element, then distributing (scattering) forces to global lists. In this same sense, the processors' set of elements can be thought of as a super element, with all degrees of freedom considered internal except those on processor boundaries. The exception here is that since boundary nodes are duplicated on each processor occurrence, the gather step of bringing global quantities to the processor is omitted. The interprocessor communication activity is done once in scattering the partial forces of boundary nodes to each duplicate node on other processors. Of course, the global minimum timestep selection is across all processors, and contact is the major exception.

One challenge in parallel algorithm development is the subdivision of a mesh to satisfy two requirements: (1) the processors should have roughly an equal amount of computational work and (2) the communication of data between processors should be minimized. Several methods for subdividing meshes have been developed recently using algorithms evolving from graph theory, grid refinement methods, spectral methods, and inertial-based methods. A large collection of mesh partitioning methods have been developed and collected by Hendrickson and Leeland and provided with the software package, CHACO (Hendrickson and Leeland, 1993). Preprocessing software has been designed at LLNL for mesh subdivision which uses the CHACO software product (Procassini et al, 1994). The finite element mesh, a list of nodes and the element-to-node connectivity, must be transformed into a list of element-to-element connections, the dual mesh, for partitioning algorithms. O(number of elements) algorithms have been developed here to produce the dual mesh for CHACO input. The result from CHACO algorithms is a list with elements assigned to processors. This data is used in a third preprocessing step to generate nodal assignments to processors, and the list of shared nodes in each processor. Figure 1 illustrates the geometrical partitioning of an automobile mesh composed of 125 materials and a mixture of solid, shell, and beam elements. While CHACO has been extremely valuable to getting an operational capability, and to evaluate seven different decomposition schemes, this task is currently done on a single off-line processor with very large memory, most algorithms are so memory intensive as not to scale well, even to a million elements. The RSB (Recursive Spectral Bisection) method of decomposition has received the most work at LLNL, across multiple groups, and will probably lead to the first decomposition algorithm widely shared among developers here for very large (greater than one million element) applications.

Contact has always been one of the strong points for DYNA3D (Hallquist and Goudreau, 1985). Contact algorithms are designed to provide an interface pressure to prevent surface nodes of a material (shell or solid) from penetrating through the surface of an adjacent material (shell or solid). This requires, (1) for each node on the surface, find the closest node in the remaining set of surface nodes; (2) for pairs of closest nodes, look up the element surface patches for one, and determine if the other node has penetrated; and (3) for a penetrated node, calculate the restoring forces determined by a penalty on the penetrating gap. A parallel contact method must localize the search step in order to avoid the communication associated with a global search. The partitioning for contact calculations is based on the geometry of the surfaces rather than the element volumes. An optimal partitioning for the contact surfaces is usually different than that for elements. For primarily local contact applications, ParaDyn uses a rather straight forward parallelization of DYNA's suite of contact algorithms. Arbitrary contact is implemented in DYNA3D by the algorithm of Whirley and Engelmann (1993). Bucket sorts impose a geometric grid of buckets over the surface node positions. In the parallel algorithm (Schauer,1990) the buckets along the boundary of processors are duplicated. The partitioning for contact assumes processors are arranged along a line in one direction on the mesh, usually the longest direction (like that logically orthogonal to which bandwidth is minimized for a direct equation solver). Surface node coordinates along this direction are sorted and evenly allocated to the processors. At each time step, the nodal positions and velocities are sent from the element partition to the contact partition and after the contact is calculated, the nodal contact forces are communicated back to the element partition. For arbitrary contact, the contact surfaces can be repartitioned as needed based on the dynamics of the contact surfaces. The extra overhead of these two concurrent decompositions explains why crash dynamics is the one application of DYNA3D, that while suitable for parallel computing, has not scaled well enough to be a true candidate for MPP, much like the direct equation solvers of NIKE3D.

The conference presentation will review a broader suite of applications. Attached here in Figure 2 is a gray-scale rendition of a one million zone quadrant of a hexagonally spaced set of buried explosive charges directing their combined effect on a more deeply buried cavity. This problem has been successfully decomposed, and run on 32, 64, and 128 processors of our Meiko CS-2, and just this week on the Cray T3D. It also fits in the 8GB shared memory of our DEC Symmetric Multiprocessor (SMP). The 32 processor Meiko took about an hour vs six hours on the DEC. The decomposition took one hour on the DEC, and is not yet fully optimized for memory or performance. The decomposition software currently is the memory limiting operation. The problem execution scaled exceptionally well for the 64 and 128 processor runs, this being the more stringent scaling test of fixed problem size

over a range of processors. For 32, 64, and 128 T3D processors, problem times were 5700, 3000, and 1700 seconds, respectively. This problem has no slidesurfaces, and is representative of what is maximally efficient at this time.

In the last few years the crashdynamics of automobiles for design and regulatory barrier and multivehicle interaction has been a major emphasis, including parallel contact-impact. But currently the emphasis is returning to nuclear weapon safety, and conventional weapons effects. Thus the port of DYNA3D to new DOD high performance computing platforms, and the applicability of earth and armor penetration, as part of a general interest in large deformation inelastic mechanics, is a current priority. Inelastic response of buried structures will exercise concrete models developed as part of our nonlinear earthquake studies (Govindjee,1995), and (McCallen,1993).

References:

Goudreau, G. L., and Hallquist, J. O. (1982). "Recent developments in large scale finite element lagrangian hydrocode technology." Computer Methods in Applied Mechanics and Engineering, V 33, N1-3, 725-757.

Govindjee, S., Kay, G.J., and Simo, J.C., (1995). "Anisotropic Modelling and Numerical Simulation of Brittle Damage in Concrete," International Journal for Numerical Methods in Engineering, V 38, 3611-3633

Hallquist, J. O., and Goudreau, G. L. (1985). "Sliding interfaces with contact-impact in large-scale lagrangian computations." Computer Methods in Applied Mechancis and Engineering, V 51, N1/3, 107-137.

Hendrickson, B., and Leland, R. Technical Reports SAND92-1460 (1992) and SAND 93-2339 (1993), Sandia National Laboratories, Albuquerque, New Mexico.

Hoover, C. G., De Groot, A. J., Maltby, J. D., and Whirley, R. G. (1994). "ParaDyn: New Generation Solid/Structural Mechanics Codes for Massively Parallel Processors". Engineering Research, Development and Technology FY93, Lawrence Livermore National Laboratory, UCRL 53868-93.

Hoover, C. G., De Groot, A. J., Maltby, J. D., and Procassini, R. J. (1995). "ParaDyn - DYNA3D for Massively Parallel Computers." Engineering Research, Development and Technology FY94, Lawrence Livermore National Laboratory, UCRL 53868-94.

Logan, R. (1993). "Crashworthiness: planes, trains, and automobiles." Energy and Technology Review, Computational Mechanics at Lawrence Livermore National Laboratory, G. Goudreau, ed.

Maker, B. N., Ferencz, R. M., and Hallquist, J. O. (1991). "NIKE3D: A nonlinear, implicit, three dimensional, finite element code for solid and structural mechanics." Lawrence Livermore National Laboratory, Livermore, California, UCRL-MA-105268.

McCallen, D. B., and Goudreau, G. L. (1993). "Large-scale computations in analysis of structures." Lawrence Livermore National Laboratory, Livermore, CA, UCRL-JC-115223.

Procassini, R. J., De Groot, A. J., and Maltby, J. D. (1994). "PARTMESH: partitioning unstructured finite element meshes for solution on a massively parallel processor." User Manual, Lawrence Livermore National Laboratory, Livermore, California UCRL-MA-118774.

Schauer, D. A., Hoover, C. G., Kay, G. J., Lee, A. S., and De Groot, A. J. (1996). "Crashworthiness simulations with DYNA3D." Proceedings of the Transportation Research Board 75th Annual Meeting, Washington D.C.

Whirley, R. G., and Engelmann, B. E. (1993). "DYNA3D: a nonlinear, explicit, three dimensional finite element code for solid and structural mechanics." Users manual, Lawrence Livermore National Laboratory, Livermore, California, UCRL-MA-107254, Rev. 1.

Whirley, R. G. and Engelmann, B. E. (1993). "Automatic contact in DYNA3D for vehicle crashworthiness." Lawrence Livermore National Laboratory, Livermore, California, UCRL-53868-03, 2-29.

*This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

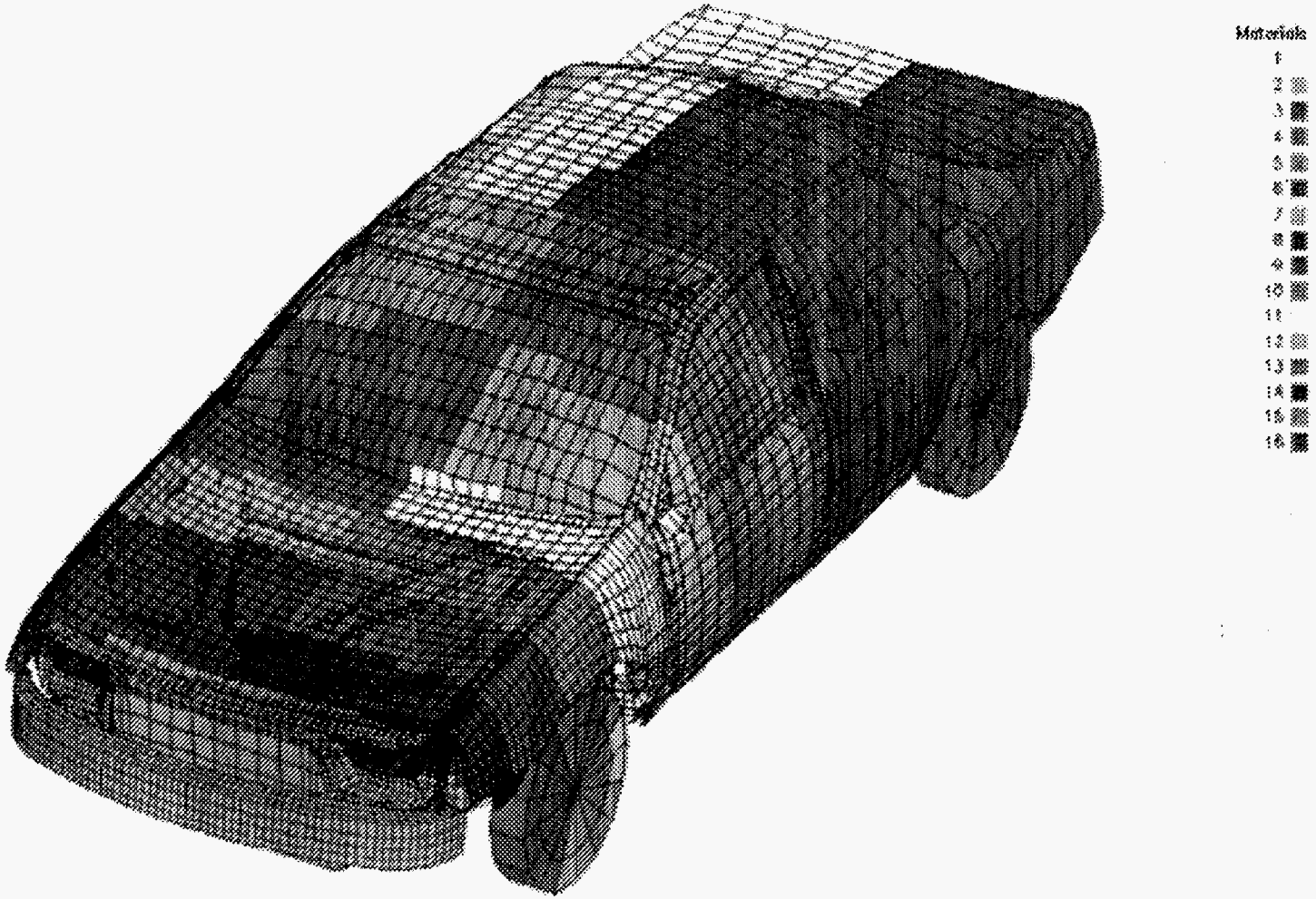


Figure 1:

Mesh decomposition of large crashdynamics model of an automobile, color coded to distribute the 27,000 elements over 16 processors.

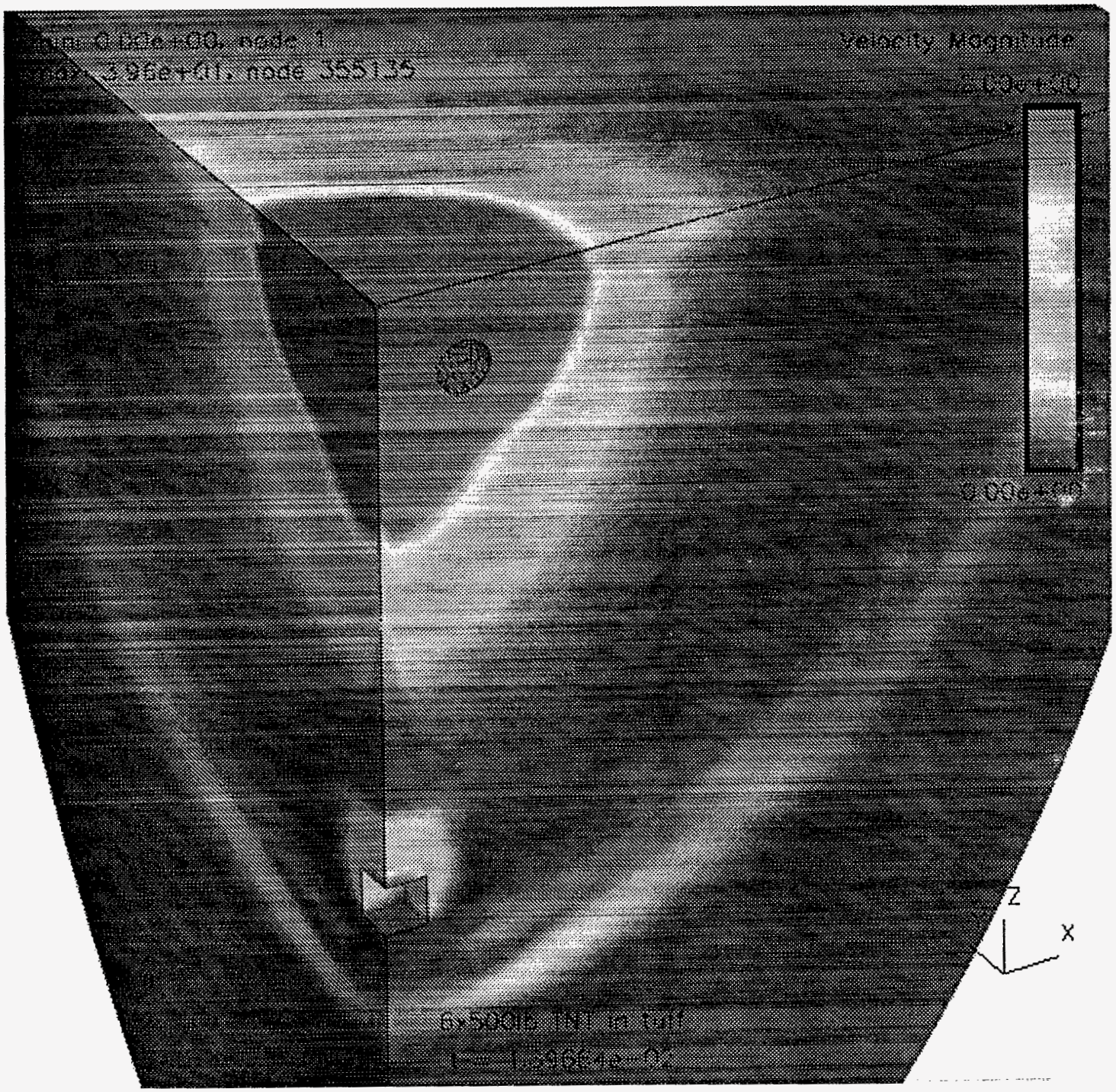


Figure 2:

One million zone model of underground effects blast calculation of hexagonally spaced buried charges propagating through homogeneous tuff toward a more deeply buried cavity. Problem run on DEC alpha SMP single processor, and 32, 64, and 128 processors of the Meiko CS-2 and Cray T3D.

Technical Information Department • Lawrence Livermore National Laboratory
University of California • Livermore, California 94551