PARALLEL ALGORITHM FOR TRANSIENT SOLID DYNAMICS SIMULATIONS USING FINITE ELEMENTS AND SMOOTHED PARTICLE HYDRODYNAMICS

Stephen W. Attaway, Bruce A. Hendrickson, Steven J. Plimpton,
Jeffrey W. Swegle, David R. Gardner, Courtenay T. Vaughan
Sandia National Laboratories
Albuquerque, NM

ABSTRACT

An efficient, scalable, parallel algorithm for treating contacts in solid mechanics has been applied to interactions between particles in smooth particle hydrodynamics (SPH). The algorithm uses a dynamic processor load balancing scheme for the contact and SPH portion of the calculation.[1]

INTRODUCTION

Combining finite elements and smooth particle hydrodynamics is attractive for simulations where part of the problem is undergoing deformations that are larger than the finite element method can accommodate, Swegle and Attaway(1995), and Tieszen, Attaway, Swegle and Slavin (1996). Implementing a hybrid mesh/particle model on a massively parallel computer poses several challenges. First is to simultaneously parallelize and load-balance the mesh and particle portions of the computation. Secondly challenge is to efficiently detect contacts that occur within the deforming mesh and between mesh elements and particles as the simulation proceeds. In our approach, we use three different parallel decompositions, a static one for the finite element analysis, and two different dynamic decompositions for SPH particles and contact detection.

The contact-detection problem poses interesting challenges for efficient implementation of a solid dynamics simulation on a parallel computer. The finite element mesh is typically partitioned to allow each processor ownership of a localized region in the finite element mesh. This mesh partitioning is optimal for the finite element portion of the calculation, since each processor must communicate only with the few connected neighboring processors that share boundaries with the decomposed mesh. However, contacts can occur between surfaces that may be owned by any two arbitrary processors. Hence, a global search across all processors is required at every time step to search for these contacts. Load-imbalance can become a problem, since the finite element decomposition divides the volumetric mesh evenly across processors, but typically leaves the surface elements unevenly distributed.

---

1. This work performed at Sandia National Laboratories supported by the U. S. Department of Energy under contract DE-AC04-94AL8500

DISCLAIMER

Smoothed particle hydrodynamics (SPH), Gingold and Monaghan (1982), Swegle, et al (1994), is a gridless method that computes spatial gradients of various quantities as a sum of pairwise interactions between a particle and its neighbors. The advantage of gridless methods like SPH is that they can handle large deformations that are typical of fluid like motion. During each time step, a search is performed to determine the neighboring SPH particles that will be included in the spatial gradient calculation. SPH elements can be added to finite element programs by treating the SPH interpolation points as special element types. Coupling of the SPH particles to the finite element mesh is achieved using a contact algorithm, Attaway, Heinstein, and Swegle (1994).

Developing an efficient parallel algorithm for SPH is difficult, because the search for neighboring SPH particles can be global in nature. Load-imbalance can also be a problem for the SPH, since the geometry can change a great deal during the problem simulation. We use a different decomposition scheme for assigning surface elements to processors than is used for the volumetric finite element mesh. The algorithm uses a static decomposition for the finite element mesh, a dynamic decomposition for determining the contacting surfaces, and an adaptive decomposition scheme for the SPH portion of the calculation. We found, in practice, that the contact algorithm is scalable to hundreds of processors. In the rest of this short paper, we will describe the basics for the dynamic decomposition scheme and outline the steps required to parallelize the finite element, contacts, and SPH portions of the algorithm.

TIMESTEP FOR TRANSIENT DYNAMICS

The basic steps for a single timestep for a combined finite element/ SPH calculation with contacts are outlined below:

1. Compute internal FE forces assuming no contacts
2. Compute internal SPH forces assuming on contacts
3. Predict motion by integrating equation of motion
4. Detect mesh/mesh and particle/mesh contacts
5. Correct motion using penalty force between contacts
6. Update the position of mesh and particles

In step (1), the FE portion of the timestep is performed. Within a single timestep in a explicit timestepping scheme, each mesh element interacts with only the neighboring elements it is connected to with the topology of the FE mesh. These kinds of FE calculations can be parallelized straighforwardly. The key is to assign each processor a small cluster of elements such that only a handfull of neighboring processors are involved in interprocessor communications through exchange of information on the cluster boundary. A static decomposition of the elements to the processors is sufficient, since the mesh connectivity does not change during the simulation. We use a software package called Chaco, Hendrickson and Leland (1995), as a pre-processor to partition the FE mesh. With this method, the interprocesor communications is minimized and each processor has an equal number of elements. In practice, the resulting FE computations are well load-balanced and scale efficiently (over 90%) when large meshes are mapped to thousands of processors.

In step (2), the SPH portion of the timestep is performed. The details of how SPH generates the velocity gradient and the stress divergence are outlined in Attaway, Heinstein and Swegle, (1994). For parallel considerations, the key concept is that two SPH particles interact if their spheres of local support overlap. An efficient parallel implementation of SPH has two requirements: (1) the number of SPH particles per processor must be balanced, and (2) the spatial region owned by a processor must be geometrically compact, enabling neighbors to be found quickly with a minimum of communication. A dynamic decomposition is required, since the spatial density of SPH particles can change quite dramatically over time.

We use recursive coordinate bisectioning (RCB) as a dynamic decomposition technique for the SPH elements. The RCB technique assigns a simple rectangular sub-domain to each processor which contains an equal number of particles. The technique has the advantage that it is efficient to implement on a parallel computer. It also has the property that a small change in the positions of the points being balanced induces only a small change in the partitions.

RECURSIVE COORDINATE BISECTIONING

The recursive coordinate bisectioning (RCB) algorithm we used was first proposed as a static technique for partitioning unstructured meshes (Berger and Bokhari, 1987). Figure 1 shows a graphical picture of how a RCB decomposition progresses. The goal of the RCB algorithm is to divide equally among P processors the combined set of N points. The first step in the RCB is to choose one of the coordinate directions, x, y, or z. for bisectioning. For this first cut, we chose the direction that results in the sub-domains being as cubic as possible. The next task is to position the cut, shown as the dotted line in the figure, at a location which puts half the points on one side of the cut, and half on the other. This step is equivalent to finding the median of a distributed set of values in parallel.

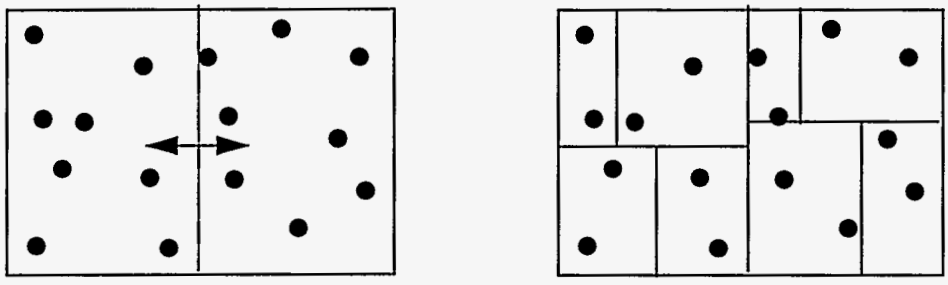


Figure 1. Left: First Cut of RCB operation. Right: Final Partitioning for eight processors.

To find the median, we use an iterative algorithm. First, we select the geometric midpoint of the box. Each processor counts the number of points it owns that are on one side of the cut. Summing this result across processors determines which direction the cut should be moved to improve the median guess. In practice, we find a suitable cut that partitions the points exactly within a few iterations. Then, we divide the processors into two groups, one group on each side of the cut. Each processor sends its points that fall on the far side of the cut to a partner processor in the other group. Likewise, each processor receives a set of points that lie on its side of the cut.

Thus, we can recurse on these steps until we have assigned *N/P* points to each processor. The final geometric sub-domain owned by each processor is a regular parallelepiped. Note that it is simple to generalize the RCB procedure for any *N* and non-power-of-two *P* by adjusting our desired "median" criterion at each stage to insure the correct number of points end up on each side of the cut.

PARALLEL CONTACTS

The contact detection in Step 4 requires determining if a node on a FE mesh or a SPH particle crosses an exterior surface on the FE mesh. This step is performed in two parts. First, a global search is performed to locate all surfaces that are near a given node or particle. A detailed check is then performed to determine the best contact surface. See Heinstein, M.W., Attaway, S.W., Mello, F. J., and Swegle, J.W.,(1993) for a more detailed description.

Parallelizing the contact detection is difficult. If we use the FE decomposition to perform the contact search, load-balance can become a serious problem. Some processor in the FE decomposition may own sections of the mesh that are completely interior. In addition, communication can overload the calculation, since a global search is required. We use the same RCB approach described above to dynamically balance the collection of contact nodes, SPH particles and contact surfaces. See Attaway, Hendrickson, et al (1996) for a more detailed description of the parallel contact algorithm.

RESULTS

Figure 2 shows the performance of the parallel contact algorithm on a scalable version of a can crush simulation. Here, the container and surface are meshed more finely as more processors are used. On one processor a 1875-element model was run. Each time the processor count was doubled, the number of finite elements was also doubled by halving the mesh spacing in a particular dimension. Thus, all the data points are for simulations with 1875 elements per processor; the largest problem was 480,000 elements on 256 processors.

As a point of reference, the grind time for a 60,000 element problem on the Cray Y-MP was T_{grind} = 41.25 μs/element/cycle. The grind time on a 32 processor problem was 40.75 μs/element/cycle on an Intel Paragon. The maximum grind time for the 256 processor problem was ~ 5μs/element/

cycle. The speed-up of the algorithm is almost linear with the number of processors.

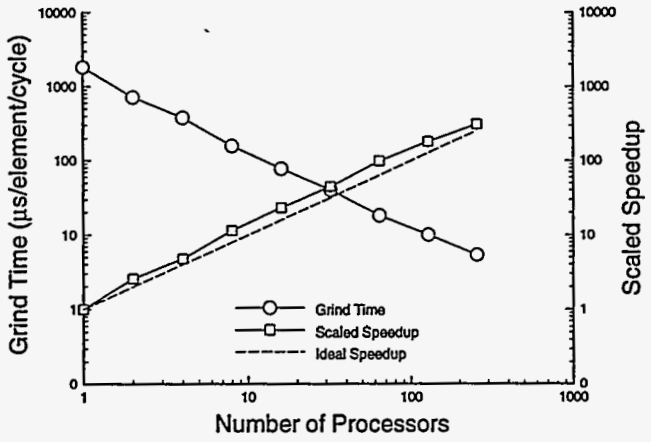


Figure 2. Grind time and parallel scaled efficiency for the scaled can crush problem.

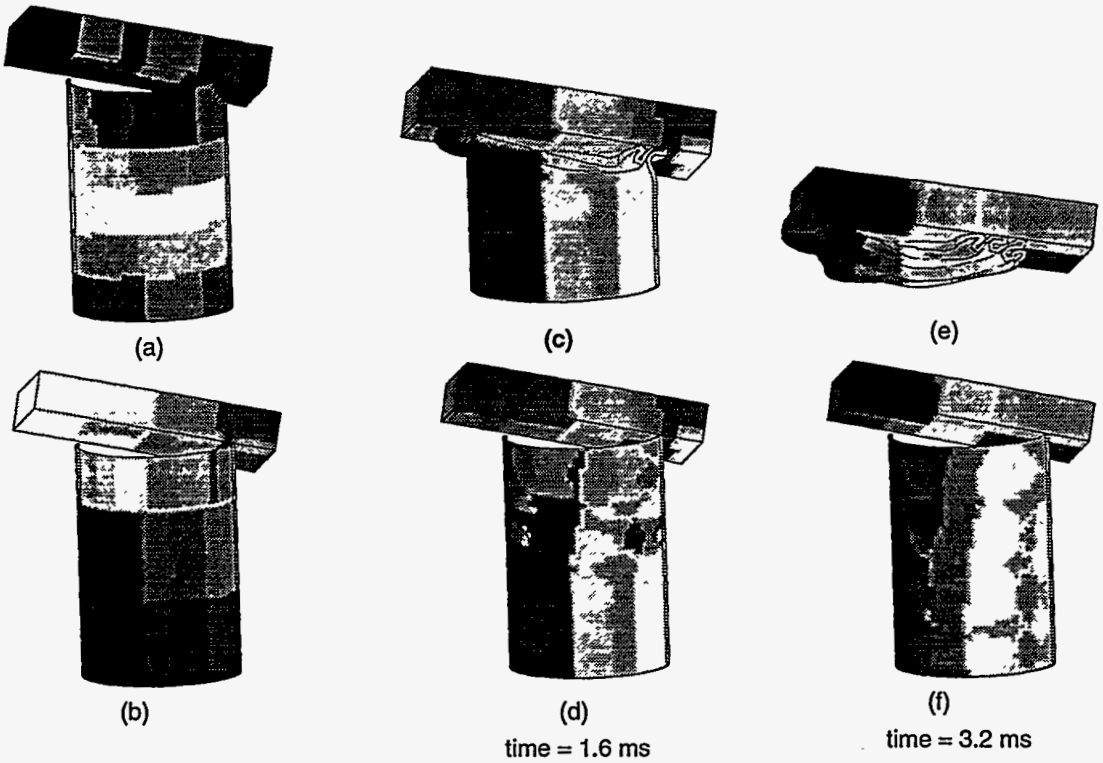


Figure 3. Decomposition generated by a) Chaco and b) at time t= 0.0, c) and d) RCB mapping for times t=1.6 ms and e) and f) t= 3.2ms. Deformed and undeformed mesh.

Figure 3 shows the Chaco and the RCB decompositions at time zero. Figure 3a shows a plot coded according to the processor that owns the element. Figure 3b shows a color coded plot where the colors correspond to which processor owns the surface. Note that the CHACO and RCB decompositions are not the same even at time zero. In fact, the top of the mesh is assigned to processor zero in the

CHACO mesh, while in the RCB decomposition, the contact surfaces in this same region are assigned to processor 31.

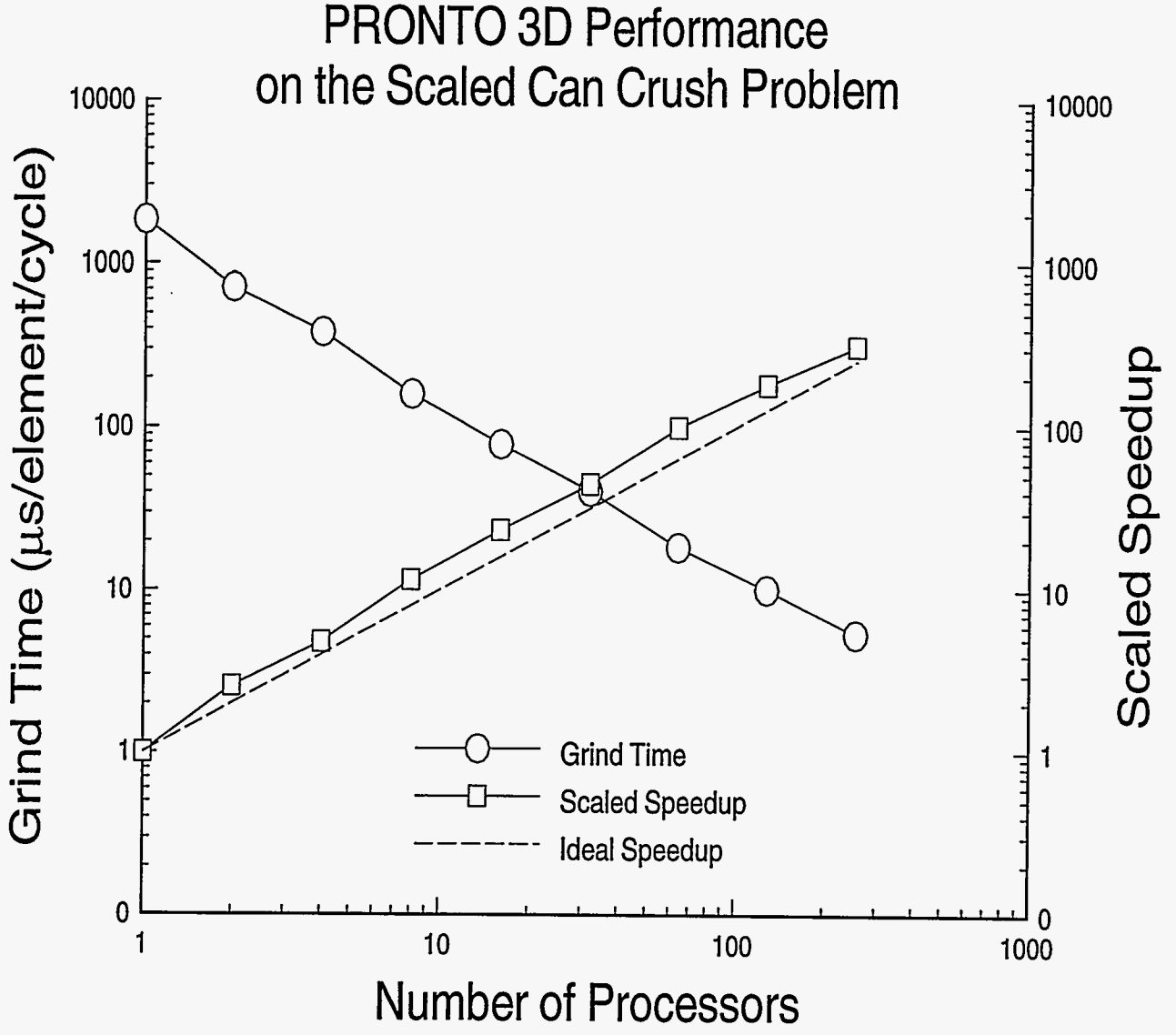
In Figure 3 d and e, the RCB decomposition at time t=1.6 ms and t = 3.2 ms is shown mapped onto the deformed and undeformed shapes. The plots show how the surfaces are locally mapped to a given processor. Some of the surfaces that start on processor 0 at the start of the problem, remain on that processor for the duration of the calculation. While other surfaces (i.e. the top of the block) are mapped onto processor zero as the calculation progresses.

SUMMARY

In summary, we have outlined a parallelization strategy for transient dynamics simulations that uses three different decompositions within a single timestep: 1) a static FE-decomposition of mesh elements; 2) a dynamic SPH-decomposition of SPH particles; 3) and a dynamic contact-decomposition of contact nodes and SPH particles. The overhead cost of such a scheme is the cost of moving mesh and particle data between the decompositions. This cost turns out to be small in practice, leading to a highly load-balanced decomposition in which to perform each of the three major computational stages within a timestep.

REFERENCES

1. Attaway, S. W., Heinstein, M.W., and Swegle, J. W. (1994) "Coupling of smooth particle hydrodynamics with the finite element method," Nuclear Eng. Design, 150, pp 199-205.
2. Attaway, S. W., Hendrickson, B., Plimpton, S., Gardner, D., Vaughan, C., Heinstein, M., and Perry, J., (1996) "Parallel contact detection algorithm for transient solid dynamics simulations using PRONOTO3D", in Proc. ASME Intl. Mech. Eng. Congress & Exposition, ASME, November.
3. Berger, M.J. and Bokhari, (1987) "A partitioning strategy for nonuniform problems on multiprocessors", *IEEE Trans. Computers*, C-36, pp 570-580.
4. Gingold, R.A., and Monaghan, J. J., (1982) "Kernel estimates as a basis for general particle methods in hydrodynamics, J. Comp. Phys., 46, pp 429-453.
5. Heinstein, M.W., Attaway, S.W., Mello, F. J., and Swegle, J.W.,(1993) "A general-purpose contact detection algorithm for nonlinear structural analysis codes," SAND92-2141, Sandia National Laboratories, April.
6. Hendrickson, B, and Leland, R, (1995) "The Chaco user's guide: Version 2.0," SAND94-2692, Sandia National Labs, Albuquerque, NM, June.
7. Swegle, J.W.; Attaway, S.W.; Heinstein, M.W., and Hicks, D.L. (1994) "An Analysis of Smoothed Particle Hydrodynamics," SAND 93- 2513, Sandia National Laboratories, Albuquerque, NM.
8. Swegle, J.W. and Attaway, S.W., (1995) "On the feasibility of using smoothed particle hydrodynamics for underwater explosion calculations, Comp. Mech., Vol. 17, pp 151-168.
9. Tieszen, S.R., Attaway, S.W., Swegle, J.W., and Slavin, A.M., (1996)"Transient, High-Speed, Liquid Impact into Soft Soil, Computer Modeling and Simulation in Engineering, Vol.1, pp 79:106,


PRONTO 3D Performance
on the Scaled Can Crush Problem
Grind Time (µs/element/cycle)
Scaled Speedup
Number of Processors
10000
1000
100
10
1
0
1
10
100
1000
Grind Time
Scaled Speedup
Ideal Speedup