UCRL-JC-128918
PREPRINT

Future Algorithm Research Needs for Partitioning in Solid Mechanics and Coupled Mechanical Models

C. G. Hoover
A. J. DeGroot
R. J. Sherwood

This paper was prepared for submittal to the
AHPCRC Workshop on Unstructured Mesh Generation and Partitioning
Minneapolis, MN
October 6-7, 1997

October 6, 1997

Lawrence Livermore National Laboratory

DISCLAIMER

Future Algorithm Research Needs for Partitioning in Solid Mechanics and Coupled Mechanical Models

C. G. Hoover, A. J. De Groot, and R. J. Sherwood
Methods Development Group and Signal/Image Processing Research Group
Lawrence Livermore National Laboratory
Livermore, CA 94550

Exceptional progress has been made in mathematical algorithm research leading to optimized mesh partitions for the highly unstructured grids occurring in finite element applications in solid mechanics. Today another research challenge presents itself. Research is needed to include boundary conditions into the algorithms for partitioning meshes. We describe below two methods we use currently to accomplish this and propose a more general approach be developed which would apply to our problems today as well as to the coupled models we envision for the future. Finally, we suggest research be considered that would incorporate partitioning methods into parallel mesh generation.

Contact boundary conditions

In solid mechanics, contact boundary conditions prevent the interpenetration of surfaces at material interfaces and can be characterized by whether the problem involves predictable motion of surfaces or whether large deformation and motion on the grid results in arbitrary interactions among the surfaces in the problem. In the first case contact is treated by defining pairs of surfaces which are expected to interact with each other and the search for material interpenetration can be limited to neighborhoods on these surfaces. We refer to this as *local* contact. The more general case occurs for problems in which the motion of the surfaces is *arbitrary.* A ball rolling on a plane is an example. In this case a localization technique, referred to as *bucketing*, is used. A novel aspect of our original implementation of a parallel algorithm for arbitrary contact was the notion of using more than one partitioning for the problem, one for the finite-element mesh and a second one for the contact boundary conditions.

Conceptually, there are several useful ideas for implementing parallel algorithms for local contact. Each of these methods involves developing techniques for partitioning the boundary condition and then generating the data structures for connecting the contact boundary partition with the partitioning for the full mesh. We describe one of these ideas for treating local contact. This method allocates a contact surface pair entirely into one processor and then distributes all surface pairs among as many processors as possible.

We use the METIS[1] software to partition our meshes. The new idea we have developed to treat the surface-pair allocation to processors is based on the notion of condensing the contact into a *superelement* containing all of the elements with a facet on the surface pair and assigning appropriate vertex and edge weights to each superelement. Then the graph for the mesh is replaced by a graph with the superelements and the remaining elements in the mesh, and the result is partitioned with METIS. This is a very advantageous approach because it automatically finds an optimal load balance for the two partitions needed for the problem. Another advantage is that all of the required shared node communication for both the contact forces and the deformation forces is known in advance at this preprocessing step.

The steps in the arbitrary contact algorithm implemented in the DYNA3D[2] program include: (1) an algorithm for generating the faces on every surface in the problem domain, (2) bucketing methods to localize the search for contact onto a coase grid, (3) a local search to find contact nodes and penetrated facets, and (4) contact force calculation. Parallel arbitrary contact is treated in ParaDyn, the parallel version of DYNA3D, with a geometric partitioning of the contact surfaces into one-dimensional strips along the longest axis on the problem domain. The strip widths are adjusted to roughly equalize the number of contact buckets allocated to a strip and each strip is associated with a processor. Because the contact is arbitrary new buckets are computed every 10 steps during the calculation. At these same times, the partitioning strips used in the parallel algorithms are recalculated and the contact buckets are redistributed among the processors. The use of these techniques localizes the contact search and load balances the parallel calculation.

Results demonstrating these methods will be presented.

Figure Captions:

Figure 1. This is a metal forming application in which a spinning plate is formed by two rollers of different size and shape. The shading shows the partitioning for four and eight processors with the surface ring under the rollers fully contained in one processor.

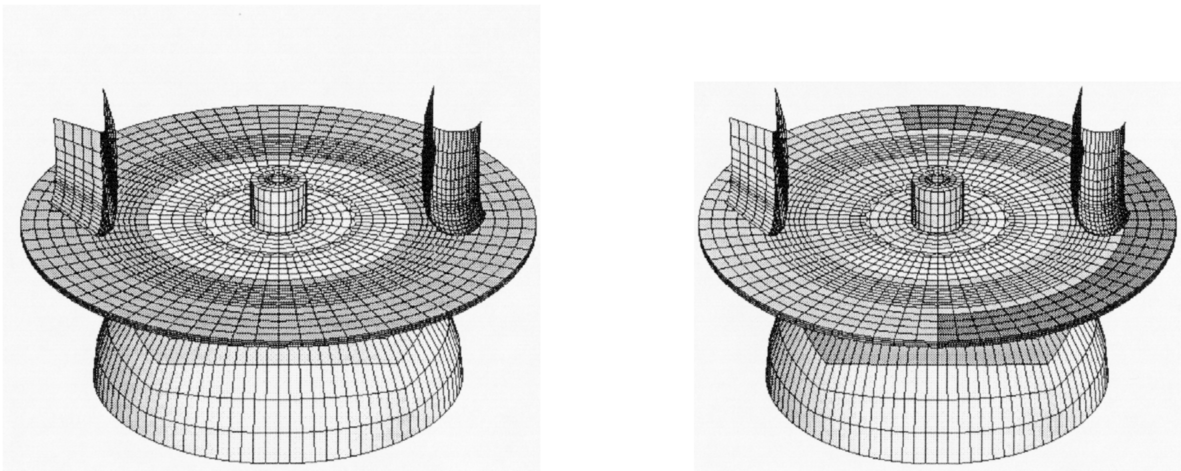
Figrue 2. This shows the parallel performance for a shipping container application. The problem input included 50,000 elements, 27 contact surfaces, 28,000 contact-related elements, and 3,200 elements in the largest contact surface pair. As a result of combined mesh/contact partitioning, the optimal number of processors for the problem changed from 8 to 32.

References:

1. Karpis, G. and Kumar, V., *METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System* , University of Minnesota, Department of Computer Science, available via WWW at http://www.cs.umn.edu/~karpis.
2. Whirley, R.G. and B.E. Engelmann (1993), *DYNA3D: A Nonlinear, Explicit, Dimensional Finite Element Code for Solid and Structural Mechanics—User*

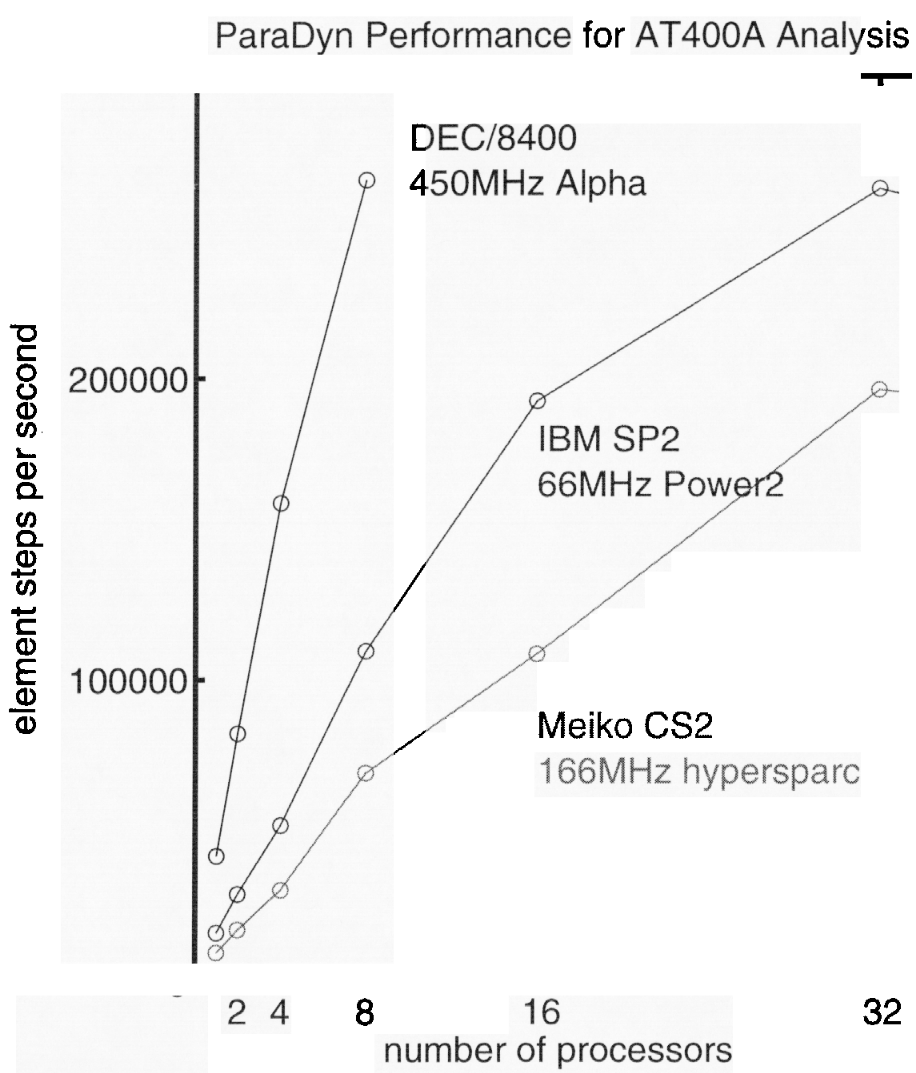
Manual, Lawrence Livermore National Laboratory, Livermore, California, UCRL-MA-107254 Rev. 1.

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.



8 Processors

Figure 1


ParaDyn Performance for AT400A Analysis
DEC/8400
450MHz Alpha
IBM SP2
66MHz Power2
Meiko CS2
166MHz hypersparc
element steps per second
200000
100000
2
4
8
16
32
number of processors

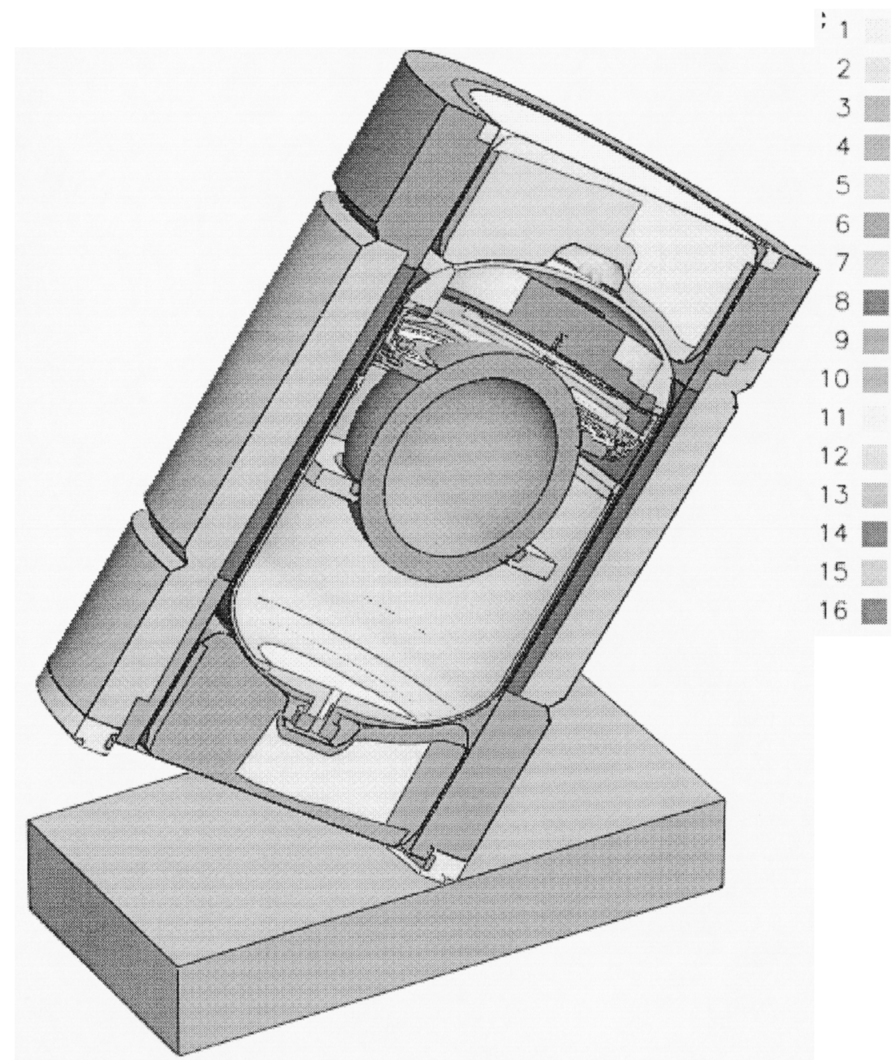

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Figure 2

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