MPP Execution of Implicit Mechanics with 10M or More Elements

Dr. C. Cleve Ashcraft, Roger G. Grimes, and Dr. Robert F. Lucas

Livermore Software Technology Corporation,

Livermore, CA, USA

Summary:

LS-DYNA models for Implicit Mechanics are getting larger and more complex. We are continually seeing models where the linear algebra problems in Implicit Mechanics have 3 to 5 million elements and know of at least one that is nearly 30M elements. It is these very large linear algebra problems that distinguish the computer requirements for Implicit Mechanics. This paper will present a study of the performance of the MPP implementation of implicit mechanics in LS-DYNA examining such issues as performance, speed-up, and requirements for computer configuration.

Keywords:

LS-DYNA, Implicit, MPP

1 Introduction

LS-DYNA has implicit technology integrated with its explicit technology. Implicit technology provides the capabilities to perform transient analyses with larger time steps as well as many linear analyses including vibration computations. As problems get larger, implicit distinguishes itself from explicit in that it has very different computer requirements. The differences are really highlighted by the requirement to factor matrices involving the global stiffness matrix. As we move to larger problems we move to larger computers which, nowadays, are MPP computer clusters with multiple cores per node.

This paper present a study of implicit requirements and performance on an MPP compute cluster housed at the Livermore offices of LSTC. We will try to set expectations of performance and give guidelines for computer hardware requirements.

2 The Test Environment

For the first test problem we are using is a 10M element version of the National Center for Automotive Crash Silverado model. LSTC refined the original model to have 10M shell elements. It has over 600 materials. The model uses both tied contact and automatic single surface contact. The resulting linear algebra problem has 60M rows and 10.6 billion nonzeroes in the factored matrix. A gravity loading is applied and one nonlinear implicit time step is executed to reach the static loading state.

The second test problem is a solid element model from the Atomic Weapons Establishment benchmark suite. This model is a simplified nonclassified version of their production analysis models. It has 6 nested cylinders held together by surface_to_surface contact. This model just uses elastic material and the constant stress solid element. The benchmark problems range from 100K to 20M elements. We are using the 4M element model. There is prescribed motion on the top and a load on the bottom.

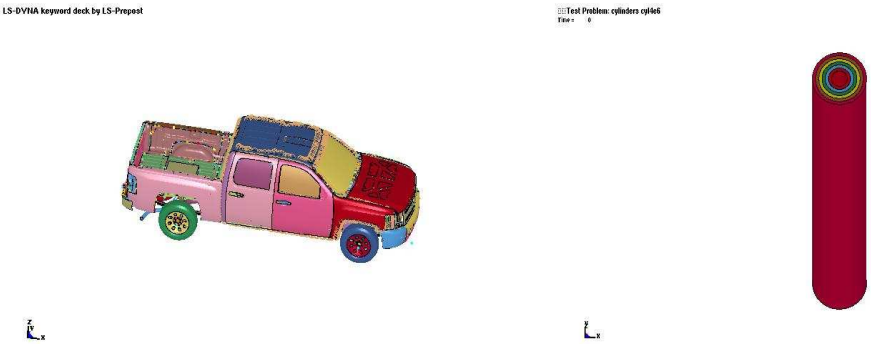

Figure 1: Test Problems

All runs were made on the fresno cluster in-house at LSTC. It has 32 compute nodes each with dual quad Intel E5520 Xeon processors at 2.27 Ghz CPUs for a total of 128 core. Each compute node has a 128 Gbyte I/O drive and 48 Gbyte of RAM.

3 Results on Silverado 10M Element Model

The following table shows the results on the 10M element Silverado Model. It contains the required memory before the linear algebra, the memory available for the linear algebra, factor wall clock time in seconds, and forward/back solve wall clock time in seconds for various combinations of compute nodes and cores per compute node

# Compute Nodes	# Cores per node	Memory pre Linear Algebra	Memory for Linear Algebra	Factor WCT (seconds)	Solve WCT (seconds)
16	1	285.0M	2703M	256.7	5.9
16	2	137.8M	1424M	135.0	3.3
16	4	82.1M	900M	81.7	2.2
16	8	Ooops!			
32	1	137.8M	1424M	135.5	3.9
32	2	82.1M	965M	79.8	2.0
32	4	42.3M	889M	58.2	1.8
32	8	Ooops!			

Table 1: Silverado 10M element MPP Performance

One can see that the memory requirements before the start of the linear algebra scales linearly as the number of cores increase. As a function of fracturing the 48Gbytes memory per node across the multiple cores the memory per core for the linear algebra decreases as well. Eventually the memory per core is too small to perform the initial model input and decomposition. The Factor time speeds up in an acceptable manner allowing for the intra-node conflicts caused by sharing nodal resources across the multiple cores. The solve time is also showing reasonable speed ups. One should note that solves may slow down due to I/O contention caused by multiple cores utilitizing the single I/O device on each compute node.

LS-DYNA also has an MPP Hybrid version. In this version we use distributed memory parallelism (MPI) across the nodes and shared memory processing (SMP) on the nodes. This keeps the memory in one more manageable unit and reduces the amount of MPI communication. Execution time is comparable between the pure MPP and the MPP Hybrid versions. But for large problems where memory considerations restricts the number of the cores that can be used per compute node then the hybrid version allows to use all of the compute cores on a compute node to reduce overall wall clock time.

# of Nodes	# of Cores	Pure MPI WCT	Hybrid WCT
16	1	256.3	227.0
16	2	135.0	130.0
16	4	81.7	85.7
16	8	Ooops!	69.5
32	1	135.5	131.3
32	2	79.8	80.6
32	4	58.2	55.5
32	8	Ooops!	51.6

Table 2: Silverado 10M element MPP and MPP Hybrid Performance

4 Results on AWE Cylinder 4M Element Model

The following table shows the results on the 4M element AWE Model. It contains the required memory before the linear algebra, the memory available for the linear algebra, factor wall clock time in seconds, and forward/back solve wall clock time in seconds for various combinations of compute nodes and cores per compute node

# Compute Nodes	# Cores per node	Memory pre Linear Algebra	Memory for Linear Algebra	Factor WCT (seconds)	Solve WCT (seconds)
16	1	30.6M	4000M	37057	1040
16	2	15.5M	2000M	20910	1185
16	4	8.0M	1000M	11137	619
16	8	Ooops!			
32	1	15.5M	2719M	19605	62.9
32	2	7.9M	1421M	10784	59.2
32	4	4.1M	Ooops!		
32	8	2.1M	Ooops!		

Table 3: AWE Cylinder 4M element MPP Performance

# of Nodes	# of Cores	Pure MPI WCT	Hybrid WCT
16	1	38849	38707
16	2	22787	23173
16	4	12580	14065
16	8	Ooops!	10062
32	1	20375	20486
32	2	11546	11899
32	4	Ooops!	7478
32	8	Ooops!	5555

Table 4: AWE Cylinder 4M element MPP and MPP Hybrid Performance

It should be noted that solid element models have a vastly increased density in the original matrix and the resulting factorization. This converts into much larger memory and time for the associated linear algebra. The large number of compute nodes allow the factorization to stay in memory to reduce the solve wall clock time. The hybrid version allows even more reduction in wall clock time because it allows the use of the additional cores per compute node.

5 Serial Memory Bottleneck

One of the *features* of the implicit implementation in LS-DYNA is a serial memory bottleneck during the Symbolic Factorization Phase. This is due to the requirement for computing the sparsity perservation ordering on the compressed graph for the entire model. We have eased this problem in the last year by having the MPI processes on a compute node cooperate on the use of available memory.

Currently LS-DYNA uses Metis. We have also evaluated other sparsity presevation ordering software. None were an improvement over Metis. We have started a long term research project for a new ordering algorithm that is both scalable in memory and time.

6 Computer Resources

6.1 Memory Requirements

As described in the previous sections, memory on a node for the MPP cluster will be a limiting factor. The amount of memory per node should be (440/P + 75)*N where P is the expected total number of cores and N is the global number of nodes in the model. The 440 comes from a simple linear fit to the memory usage for a number of problems including the two test problems in this paper.

Remember that the memory on a node not only has to hold the LS-DYNA work array whose length is given by the memory=xxxxM on the command line but also dynamic memory for MPP core utilities including Metis, the matrix assembly package and the LS-DYNA executable. The authors always use the following memory specifications on 16 and 48 Gbytes per node clusters

16 Gbytes		48 Gbytes	
Cores	Memory	Cores	Memory
1	1500M	1	4000M
2	800M	2	2000M
4	400M	4	1000M
8	200M	8	500M

The authors also recommend not using the memory2= option on the command line for implicit. The resulting memory imbalance among the processes will negatively affect the overall performance.

6.2 I/O Requirements

To keep memory to an acceptable level the factorization of the global stiffness matrix must be stored on disk for these large problems. At LSTC we have 128 Gbyte disks on each node. This allows us to easily hold the scratch files for the test problems in this paper.

One should also remember that the multiple cores on a node will contend with the single I/O resource so fast disks are also required. While not covered in this paper, we expect that this I/O contention will have a more negative effect on eigenvalue computations. For those problems fewer cores might lead to better overall performance.

Our experience to-date indicates that local disks on each node give better performance for the scratch files for the numerical factorization. This is because there are only 1 to 8 I/O streams interacting with each other on the local disk. On a global file system there will be an I/O stream for each MPI process. Such large number of I/O streams and the vast volume of data in each stream will easily overwhelm most global file systems.