Progress on GPU Implementation for LS-DYNA Implicit Mechanics

Roger G. Grimes, Dr. Robert F. Lucas, and Gene Wagenbreth

Livermore Software Technology Corporation,

Livermore, CA, USA

Summary:

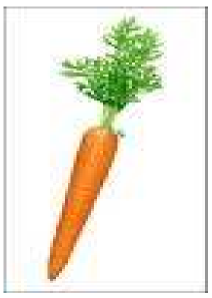
Graphics processing units (GPUs) are ubiquitous devices designed to improve the end-user experience in mass market arenas such as gaming. High-end GPUs have an order of magnitude more computing power than their hosts, and are thus attractive candidates for accelerating compute bound applications such as MCAE. This talk will present how we have extended LS-DYNA to utilize Nvidia Tesla GPUs for implicit mechanics. We will describe the target environment along with performance results on a range of benchmark problems. The performance results will illustrate when it makes sense today to utilize the GPU, and when it does not.

Keywords:

LS-DYNA, Implicit, MPP, GPU

1 Introduction

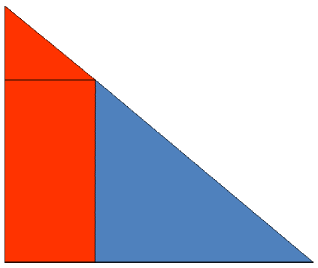
LSTC has been working on incorporating the computation power of GPUs into LS-DYNA. But that enhanced computing power does not come without limitations. GPUs usually have less memory than the host computer; the computations have to be in the form of single instruction-multiple data written in a different computer language; and the input/output bandwidth between the host and the GPU is relatively slow. On the other side of the argument the GPU has a peak computational rate that is 4 times more than the usual dual quad core CPU on the host. The GPU computational rate is the carrot in front of the donkey.



The slow transfer rate between the host and the GPU means that O(100) floating point operations have to be performed on the GPU per word of data transferred. The only computational unit in LS-DYNA that qualifies for migration to the GPU is the core computational unit of the direct factorization of the global stiffness matrix required in Implicit Mechanics. This presentation will focus on this aspect of LS-DYNA.

LS-DYNA can spend large amounts of computational time in Contact and Element Processing. These are not candidates to migrate to the GPU because there is not enough computing per data unit to overcome the slow transfer rate. It is infeasible to leave the data on the GPU and only transfer the results back to the host. Furthermore it is not practical to convert the numerous and varied algorithms for contact, materials, and elements to meet the requirements of GPU computing.

2 Direct Factorization in Implicit Mechanics



The dominant computation unit of the multifronal algorithm used for the direct factorization in LS-DYNA is the partial factorization of a real symmetric positive definite matrix. There are k columns ready to factor and l columns to be updated and passed up the computational tree. These are known as frontal matrices.This computation can represent 95% of the work for the factorization. The size of k and l vary throughout the computation and are problem dependent. The work is more concentrated for solid models compared to automotive applications.

3 The Test Environment

All of the performance numbers published in this paper are using a PC at LSTC equipped with a dual quad core Nehalem Xeon 5560 processors and 2 Nvidia Tesla boards. The host has 96 Gbytes of memory while each GPU has 2 Gbytes of memory.

Our current GPU implementation uses just one GPU per MPI rank. Since we have just 2 GPUS our testing uses either 1 or 2 MPI ranks with or without a GPU for each rank. We use the MPP Hybrid version of LS-DYNA 971 so we can make use of the additional cores on the host. The host performance numbers are always using a total of 8 cores. When the work for the partial elimination of the frontal matrix is large enough then that work is transferred and performed on the GPU. Otherwise it is performed on the host. At the top of computational tree we usually have the various MPI ranks cooperating. We have not yet reached this state of complexity with the GPU environment. We restrict the partial elimination of the top level frontal matrices to just use a single GPU. (We expect to enhance this later in 2011.)

For some problems pivoting for numerical stability is required. Pivoting is very hard to perform on the GPU due to the required programming model. When the need for pivoting is detected the partial elimination on the GPU is terminated and the host performs the computation. This is a rare event.

The first test problem is a solid element model from the Atomic Weapons Establishment Cylinder 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 nodes. We are using the 1M node model. There is prescribed motion on the top and a load on the bottom.

For the second test problem we are using is the original Silverado model from National Crash Analysis Center. It has over 600 materials and uses both tied contact and automatic single surface contact. This model has 942651 nodes and 929070 elements, most of which are shells. The resulting linear algebra problem has 5.3M rows and 1.5 billion nonzeroes in the factored matrix. Gravity loading is applied and one nonlinear implicit time step is executed to reach the static loading state.

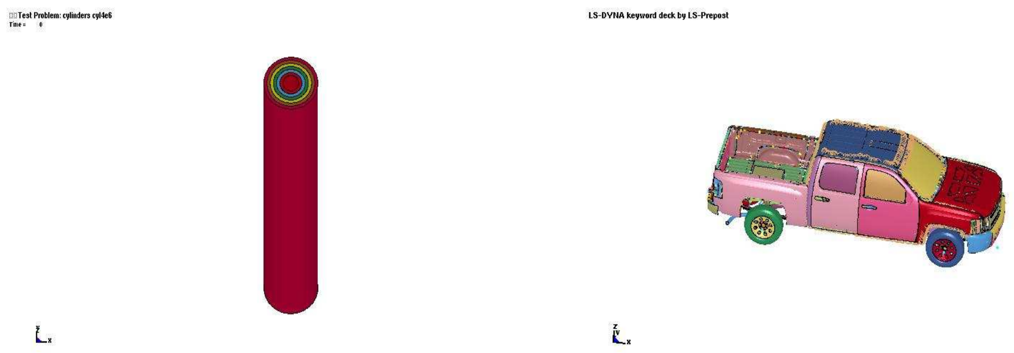

Test Problem: cylinders cyl4e6
Time = 0
LS-DYNA keyword deck by LS-Prepost

Figure 1: Test Problems

4 Results on AWE 1M Cylinder Model

The following table shows the results on the 1M node AWE Cylinder Model. It compares factor wall clock time in seconds between using GPU and just using the host computer. We use either 1 or 2 MPI ranks and a total of all 8 cores on the host. Also included is a comparison of Total Elapsed wall clock time in seconds.

# MPI Ranks	Factor WCT without GPU (seconds)	Factor WCT with GPU (seconds)	Elapsed WCT without GPU (seconds)	Elapsed WCT with GPU (seconds)
1	10111	2885	25359	9163
2	9682	2251	23986	8387

The promise of GPU computation is evident from this chart. In the one computational aspect that we have worked on the wall clock time is slashed to ¼ of the host time using 8 cores.

But that is not the entire story. The bottom line is the total elapsed wall clock time for an LS-DYNA execution. There the reduction is more like 1/3. Below is the breakdown of time for various segments of the computation

	Without GPU	With GPU
Element Processing	228	228
Factorization	21739	5900
Forward/Back Solves	2908	2908
Other	484	484
Total	25359	9163

The wall clock time for forward and back solves is limited by the speed of the I/O unit. It is providing a average transfer rate of 120 Mbytes per second while transferring 90.8 Gbytes for each of the 2 solves. Given the slow communication between the host and the GPU there is no expected benefit from using the GPU for Element Processing or any other phase of LS-DYNA.

These are the best performance numbers that can be expected. The AWE cylinder problem represents an intensely concentrated set of computations for the factorization. This concentration of computations hits the sweet spot for using the GPU. It also justifies LSTC choice of focusing solely on the matrix factorization.

5 Results on NCAC Silverado Model

Unfortunately the results are not as good for the Silverado model. This is expected. The computational work for automobile models such as the Silverado is less concentrated as for solid element models. Furthermore this is a nonlinear model with more of the computational time spent in the rest of LS-DYNA. The Factor time is for a single factorization. The simulation actually required 8 factorizations

# MPI Ranks Nodes	Factor WCT without GPU (seconds)	Factor WCT with GPU (seconds)	Elapsed WCT without GPU (seconds)	Elapsed WCT with GPU (seconds)
1	85	60	5170	5084
2	60	45	5213	5089

The first question is why is the speed-up so meager for the factorization. The computational work is less concentrated for Silverado model. The frontal matrices are smaller so less work is actually done on the GPU. And what work is performed on the GPU is not sufficiently large to hit the top computational rates. So less speed-up is expected for such models.

The second question is about the Elapsed Time. This is because 85% of the overall work is performed in element processing, contact, and the other aspects of implicit. Even with the GPU reducing the factorization time to zero there would only be a 15% reduction of time.

6 Summary

The results for using the GPU on automobile models or similar shell element based models is still very preliminary. This is our first results using such models. While we hope to improve the GPU performance on such models the results do not look good for the GPU. The slow communication interface and restrictive programming model will inhibit the use of LS-DYNA and GPUs on such models.

On the opposite end is the spectacular results on solid element models such as the AWE cylinder model. The overall speed-up of a factor of 3 can be an incredible boon to users with such models. Expectations of performance improvement should be based on where your models lie in the spectrum between automobile and solid models.