The Potential Impact of GPUs on LS-DYNA® Implicit

Roger Grimes, Robert Lucas, and Gene Wagenbreth
Livermore Software Technology Corp.

Abstract

This talk will report the on-going efforts of LSTC to study the impact of GPUs on LS-DYNA. GPUs offer very high performance computational power at the cost of importing and exporting of data between the host computer and the GPU. The GPU has restricted memory, requires programming in a special language, and suffers performance reduction for double precision arithmetic. Still GPUs appear to offer a potential speed-up of a factor of 2 to 3. Initially our study is focusing on the impact on Implicit Mechanics. The most important computational kernel for Implicit Mechanics is in the direct solution of the sparse systems of linear equations. We will focus on how one can use GPUs for this computational kernel and the probable performance improvement.

Overview of GPUs

LSTC is studying the impact of GPUs on LS-DYNA. In particular we are currently focused on the currently available NVIDIA Tesla GPU and the soon to be released NVIDIA Fermi GPU. These devices offer potential computational speed that is 15 times faster than a single core of the currently available CPU processor chips. But this additional computational speed is not free. There is the cost of stopping the computation on the host CPU, packaging and shipping the data to the GPU, actually performing the computation on the GPU, and then shipping the results back to the host CPU. The GPU has less memory restricting the data size associated with the computational kernel on the GPU. Currently the computations on the GPU have to be programmed in CUDA. Additionally the Tesla GPU does not support double precision but the Fermi GPU does.

Software Kernels in LS-DYNA

LSTC has reviewed the computational kernels in LS-DYNA to determine the likely candidates for exporting to GPUs. LSTC has already performed extensive computational performance enhancements on all of the computational kernels associated with explicit mechanics such as element processing, contact, and constraint application. Unfortunately most of computational kernels do not have enough floating point operations per unit of memory to justify migrating to the GPU. The cost of transferring the data to the GPU and the results back from the GPU is more expensive than the original computation on the host CPU.

The exception is the computational kernel for the sparse direct linear equation solver used by Implicit Mechanics. For large Implicit Mechanic problems the cost of matrix factorization is 80% to 90% of the overall cost of the LS-DYNA. This solver is based on the Multi-frontal method. At each step of the algorithm a real symmetric dense matrix is formed representing a subset of overall linear system that is ready to be processed. For those familiar with the frontal method, this is the frontal matrix. Multi-frontal uses a computational tree structure where every node of the tree requires assembly of this frontal matrix from contributions of the descendent nodes in the tree and the original matrix. Once the frontal matrix is assembled a given number of

columns are factored and the rest of the matrix is updated and passed up the tree. This partial factorization phase represents 80% to 90% of the cost of factoring the stiffness matrix for implicit mechanics. Remember that for large problems this factorization can represents 90% of the overall cost of large problems in LS-DYNA Implicit mechanics. So this one computational kernel represents approximately 70% to 80% of the overall cost of LS-DYNA Implicit Mechanics for large problems. And the computational kernel has a high computational operations to data ratio that makes it attractive to migrate to the GPU.

What We Have Done So Far

LSTC has a in-house computer with dual Nehalem Xeon 5560 CPUs and 24 Gbytes of memory. It is equipped with two NVIDIA Tesla GPUs.

At the time of the writing of this paper LSTC has coded the partial factorization computational kernel for the multi-frontal method in single precision in CUDA for the NVIDIA Tesla. This computational kernel achieves nearly 13 Gflop/sec on a single Nehalem CPU. On the Tesla we are currently achieving 130 Gflop/sec. Comparing just the computational times the Tesla is 10 times faster than the host CPU. Review of the CUDA code by NVIDIA analysts indicate some performance enhancements that could potentially increase performance to 150 Gflops or an 11 times speed-up over the host CPU.

We have an integrated direct linear equation solution test bed running on the host CPU. This test bed is using the same solver as used in LS-DYNA. Because the Tesla GPU is limited to single precision this testing is all with single precision arithmetic. (Note that we strongly recommend using double precision for Implicit Mechanics.) We are not yet running LS-DYNA in this environment. Frontal matrices large enough to justify transferring to the Tesla GPU have the partial factorization performed on the Tesla GPU. At this time frontal matrices that are too large to fit in memory on the Tesla GPU are still processed on the host CPU. Initial testing demonstrates an overall speed-up of 2.6. It should be emphasized that we are using only one of the Nehalem CPUs and only one of the Tesla GPUs for this comparison.

Next Steps

We are planning two additional steps that should be accomplished by the time of the LSTC Users Meeting in June 2010. By this time LSTC will have replaced the Tesla GPU with the new Fermi GPU. This will provide a native double precision computation on Fermi. We will recode the CUDA to use double precision. Then we will compare the double precision computational performance of the host CPU and Fermi using the established sparse direct solver test bed.

We also expect to port LS-DYNA to this computer system. We will create a version that utilizes the modified sparse direct linear equation solver. This will allow us to test the overall performance changes within the context of the entire LS-DYNA execution.

From the experience with the computational kernel for multi-frontal on the Fermi GPU we will also be able to better judge if there are any additional computational kernels that would benefit from porting to Fermi GPU. At that time we will have up-to-date information on the communication costs and memory limitations of the Fermi GPU.