A Comprehensive Study on the Performance of Implicit LS-DYNA

Yih-Yih Lin
Hewlett-Packard Company

Abstract

This work addresses four aspects of Implicit LS-DYNA's performance. First, Implicit LS-DYNA's scalability is characterized on multicore platforms. Second, the effectiveness of the GPU implementation is examined. Third, the effect of memory configuration on performance and information on how to configure optimal memory size for a system are presented. And fourth, the performance of out-of-core solutions is discussed.

Introduction

Cost per time step in Implicit LS-DYNA is much more expensive than in Explicit LS-DYNA. The major cost for implicit solution is the cost of matrix factorization in the algorithm for solving a linear algebraic equation or an eigenproblem. Matrix factorization is both very CPU and memory intensive in comparison to explicit calculation. This paper investigates four aspects of LS-DYNA implicit solution. First, that LS-DYNA can perform unsurpassable speed in implicit solutions with modern multicore servers and clusters is established by characterizing its scalability. Second, how the recent GPU implementation helps the performance of implicit solutions is investigated. Third, the effect of memory configurations on performance is exemplified by comparing performances on a 1 DPC (DIMM per Channel) system and a 2 DPC system. Additionally, information on how to configure a system with optimal memory sizes for implicit solutions is presented. And fourth, the performance of out-of-core solutions is discussed.

In this investigation, the Hybrid LS-DYNA 971 R6.0 version and the GPU-enabled Hybrid LS-DYNA R6.0.1 beta version are used. The four cylinder models of 0.5, 1, 2 and 4 million solid elements—known as the Cyl0p5e6, the Cyl1e6, the Cyl2e6, the Cyl4e6 models—are studied. And the following three HP multi-core platforms are used:

- HP ProLiant DL980: a 1-TB shared memory server, comprising eight ten-core Intel Xeon E7-4870 processors at 2.4 GHz.
- HP ProLiant SL390 cluster: a 32-node cluster interconnected by InfiniBand QDR, each node comprising two six-core, 2.93 GHz Intel Xeon X5670 processors and having 48GB memory; 16 nodes of which also configured with 3 NVIDIA TESLA M2090 6 GB GPU.
- HP ProLiant SL230 cluster: an 8-node cluster interconnected by InfiniBand FDR, each node comprising two eight-core, 2.6 GHz Intel Xeon E5-2670 processors and having 64 GB memory.

Scalabilities

As discussed later, an out-of-core solution is always slower than its corresponding in-core solution; moreover, performance of the former depends on several factors and varies greatly. For this reason, the in-core solution is used in the following discussion unless otherwise mentioned.

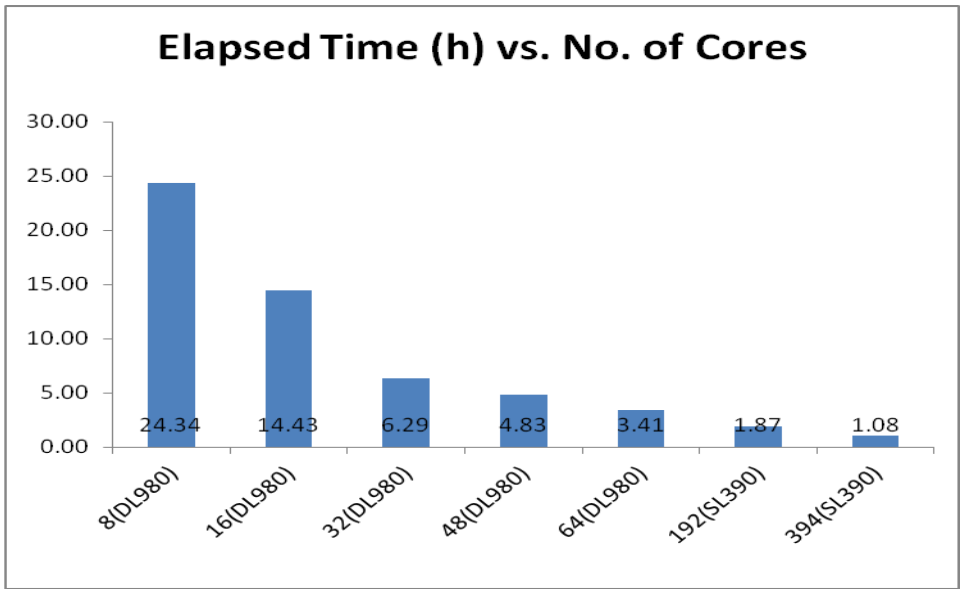

Figure 1: Elapsed times for the Cyl4e6 model

Shown in Figure 1 are the elapsed times measured with 8, 16, 32, 48 and 64 cores on the DL980, and 192 cores and 394 cores on the SL390. The corresponding scalabilities, relative to 8 cores on the DL980, are shown in Figure 2. The DL980 achieves a sublinear scalability: increasing the number of cores 8 times results in a speedup of 7 times. The SL390 similarly achieves a sublinear scalability at a greater number of cores: increasing the number of cores 2 times results in a speedup of 1.7 times. It is noteworthy that increasing the number of cores 48 times, from that of the DL980 to that of the SL390, results in a speedup of 23 times. (The Intel processors used in the DL980 and SL390 are in the same generation.) This kind of scalability is indeed the best among all commercial implicit mechanics software.

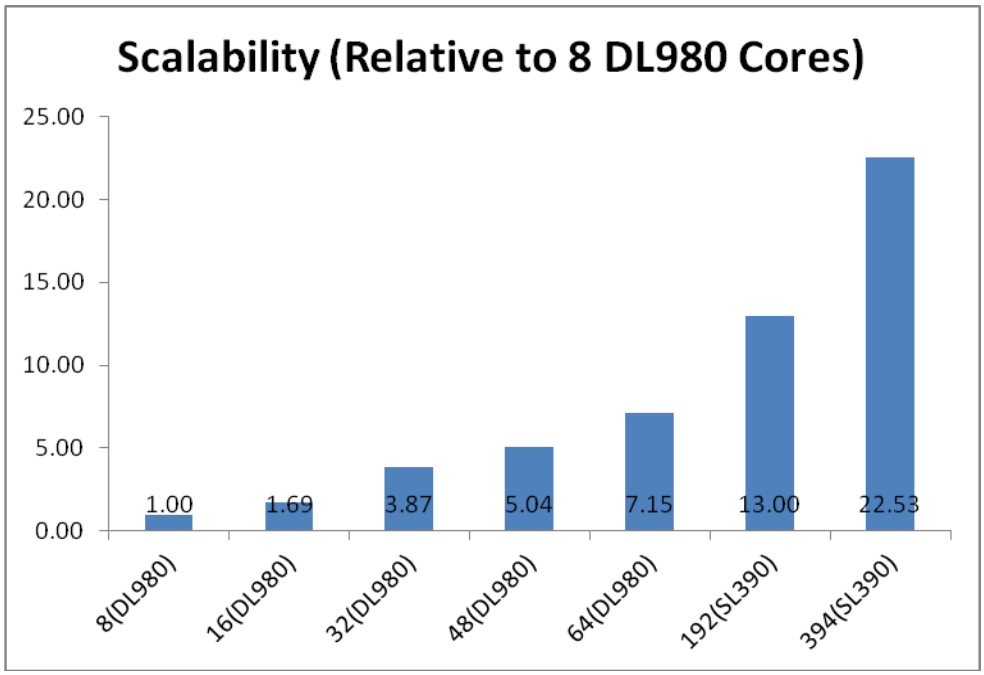

Figure 2: Scalability for the Cyl4e6 model (from Figure 1)

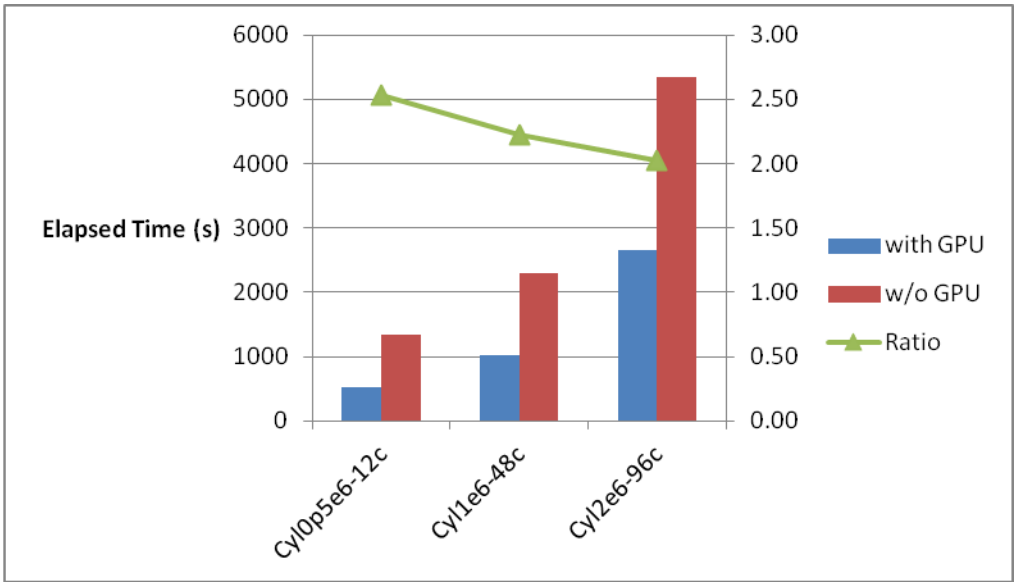

Figure 3: Comparison of elapsed times with the GPU implementation and without. The number of cores increases as the model size increases.

Effectiveness of the GPU implementation

As mentioned earlier, matrix factorization represents a major cost in an implicit solution. Because matrix factorization involves arithmetic operations on a great number of one-dimensional arrays of data, GPU can greatly accelerate its speed. Figure 3 compares the elapsed times with the GPU implementation and without for three cases: the Cyl0p5e6 model with 12 cores, the Cyl1e6 model with 48 cores, and the Cyl2e6 model with 96 cores. The GPU implementation has achieved 2 to 2.5 times speedup.

Memory Configuration

Matrix factorization is a process of continuously reading, updating and writing matrix data, which uses CPU and memory. Memory speed is relatively much slower than CPU speed. It follows from Amdahl's law that improving the speed of the slower component *memory* is effective in improving the speed of an implicit solution. The computer industry offers an array of choices on memory modules and configurations with varying speeds. Figure 4 shows that the SL230 configured with 2 DPC (DIMM per Channel) outperforms that with 1 DPC by 1.13 times for the Cyl0p5e6 model and by 1.21 times for the Cyl1e6 model. This is a result of Amdahl's law as a system configured with 2 DPC has higher memory bandwidth than the one with 1 DPC.

Configuring systems with optimal memory size within a budget requires knowledge of memory requirements for in-core solutions on typical models. The Hybrid LS-DYNA is a distributed program, in which SMP threads are first distributed across ranks, which in turn are distributed across nodes. Memory requirement is therefore distributed per rank. Figure 5 shows the in-core solution memory requirement for the Cyl2ep6 model with the number of ranks from 1 to 8, and Figure 6 shows that for the Cyl4e6 model with the number of ranks from 4 to 64.

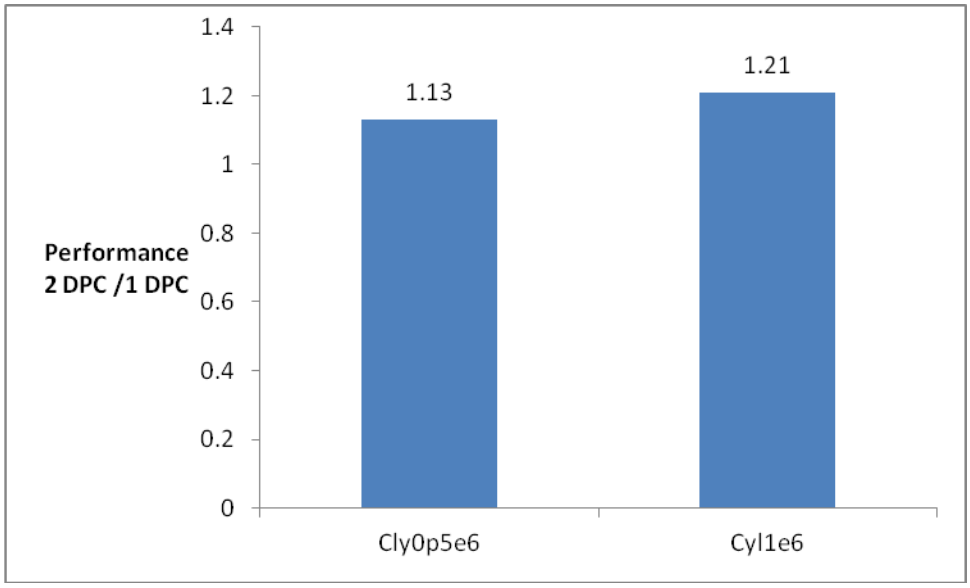

Figure 4: 2 DPC configuration outperforms 1 DPC configuration

It is noteworthy that the total memory requirement varies little with respect to the number of ranks. If the solution is run on a single server, like the DL980, the required memory size is the same as the total memory size. However, if the solution is run on a cluster of nodes with a uniform memory size, like the SL390, then the required memory size for each node is the maximal required memory times the number of ranks used in each node. Figure 7 shows required memory per node with respect to the number of ranks for the Cyl4e6 model. Note that the required memory per node decreases almost linearly with increasing number of ranks. As a result, if a cluster with a small memory size per node is used, one can solve a big implicit problem in-core by increasing the number of ranks, or equivalently by increasing the number of nodes.

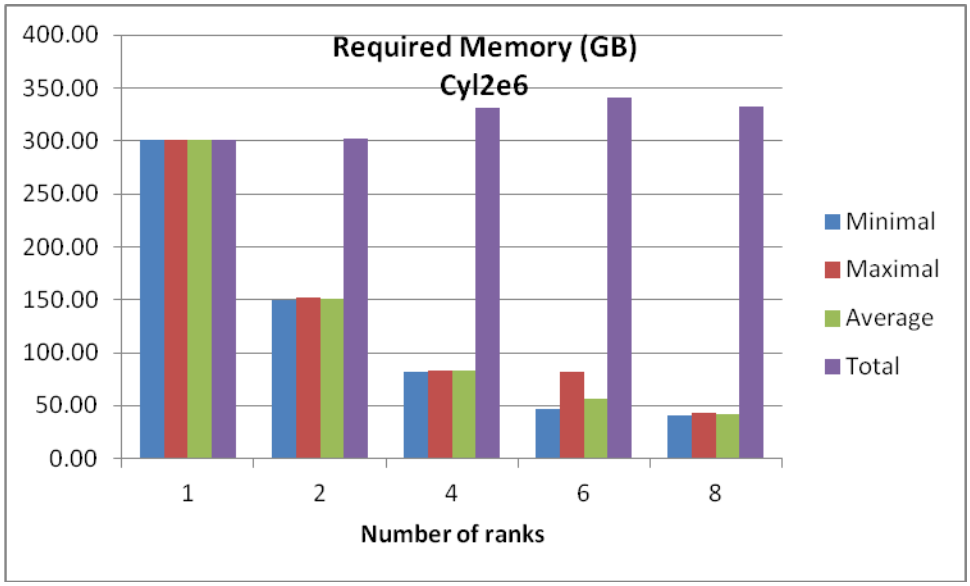

Figure 5: Memory requirement for the Cyl2e6 model

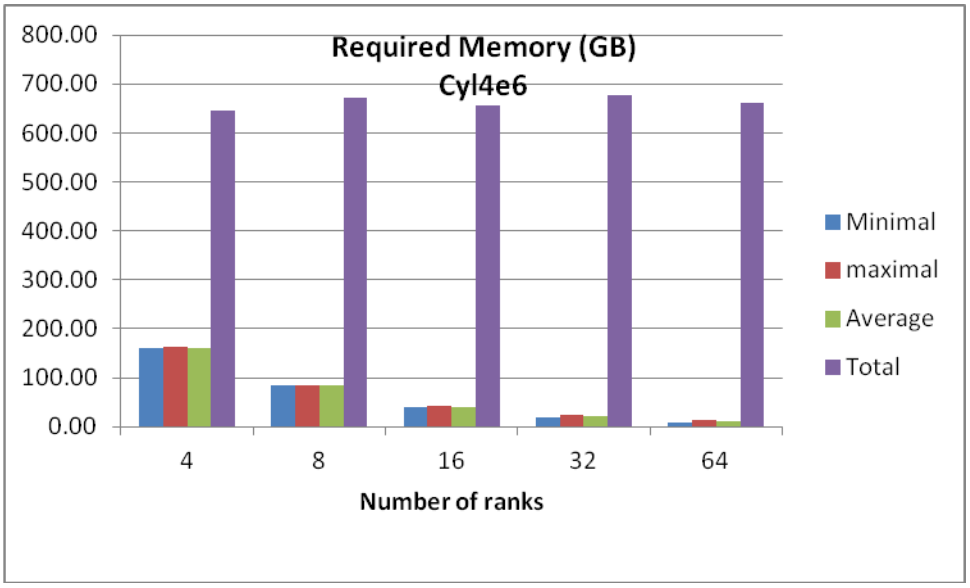

Figure 6: Memory requirement for the Cyl4e6 model

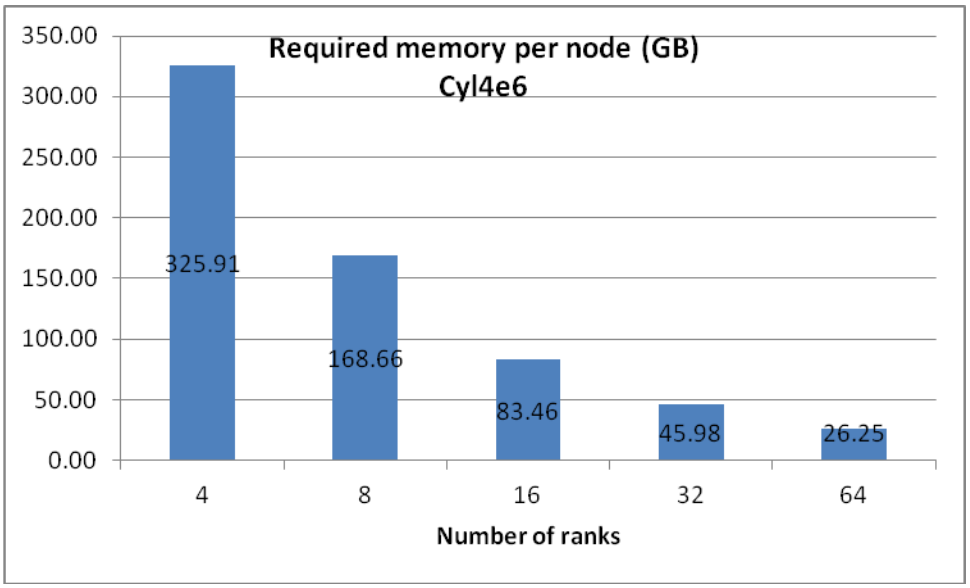

Figure 7: Memory requirement per node in a distributed cluster, using 2 ranks per node

Out-of-Core Solutions

An out-of-core solution uses both main memory and I/O to perform its memory operations while an in-core solution uses only main memory. Since I/O is slower than main memory, an out-of-core solution is always slower than its corresponding in-core solution. But factors that determine performance penalty in out-of-core solutions are many folds and complex. The following factors are some examples:

1. I/O systems with a wide range of speeds are offered by the computer industry.
2. In a shared I/O environment, the heavier the load, the slower the I/O.

3. The amount of free memory for buffering impacts the I/O speed.
4. The amount of free memory limits the amount of main memory that can be allocated for the out-of-core solution and thus limits the amount of its in-core memory operations.

Figure 8 shows the relative performance of the out-of-core solution for the Cyl2e6 model on the DL980 with three different amounts of free memory. As shown, the least free memory case (326GB) incurs a 60% speed penalty. This is a consequence of factor 3.

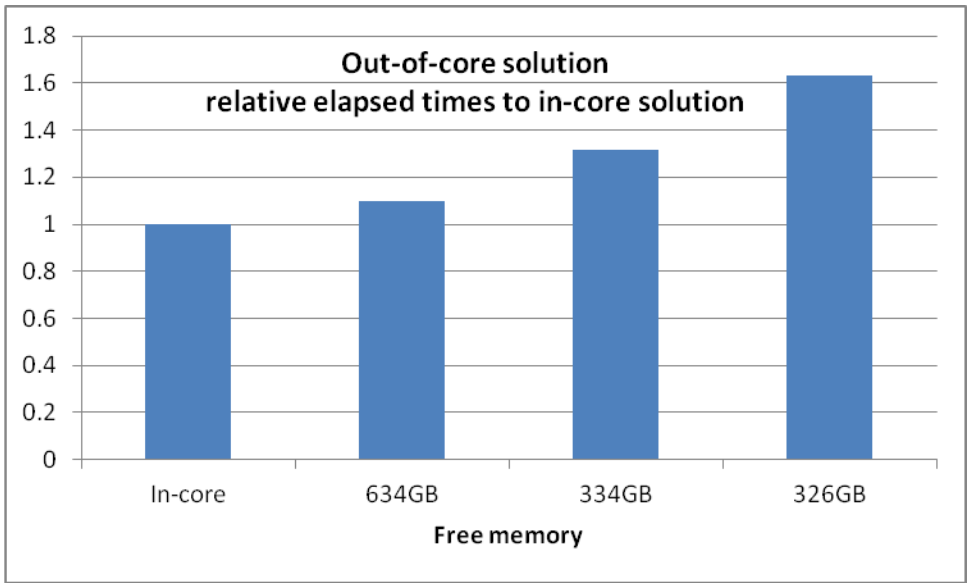

Figure 8: The amount of free memory can affect speeds of out-of-core solutions. The command line option *memory* is set at the same size for the 3 out-of-core solutions.

Conclusion

In summary, this paper establishes the unsurpassable scalability of Implicit LS-DYNA on multicore platforms. It shows that a platform configured with the GPU implantation outperforms that without more than 2 times. Furthermore, it demonstrates the importance of memory configuration by showing that a platform configured with 2 DPC outperforms that with 1 DPC more than 1.13 times. And finally, the nature of out-of-core solutions is discussed.