LS-DYNA® on Advanced SGI® Architectures

Olivier Schreiber, Scott Shaw, Brian Thatch*, Bill Tang†
*SGI Applications Engineering
†SGI Systems Engineering

Abstract

LS-DYNA's implicit solver integration with explicit software allows large time steps transient dynamics as well as linear statics and normal modes analysis. Until recently,this capability could only be run on large Shared Memory Parallel (SMP) systems,where the application had access to large memory address space of the model. Distributed Memory Parallel (DMP) implementation of LS-DYNA's implicit solver now allows the factorization of smaller mass and stiffness matrices of the decomposed problem domain by corresponding tasks in less memory. Performance enhancement through SMP processing is moreover also available in the recently introduced 'hybrid' mode. This paper demonstrates how advanced SGI computer systems, ranging from SMP servers addressing large memory space through multi-node clusters can be used to architect and accelerate solutions to meet complex analysis requirements.

Introduction

The subject of this paper is to evaluate the applicability of the SGI Octane™ III, Altix ® XE, Altix ICE, Altix UV and Altix architectures to Shared Memory Parallel (SMP), DistributedMemory Parallel (DMP) and their combination (hybrid mode) LS-DYNA implicit analyses. The strategies employed by LS-DYNA and the practical importance of such analyses are described in Refs [1] and [2]. Integrated within its explicit framework, LS-DYNA's implicit technology provides the capability to perform transient analyses with larger time steps as well as usual linear statics and modal analyses. How to best use SGI hardware is described in Ref [3].

1 Benchmark Description

The benchmarks used are identical physical problems as in Refs [1] and [2] available in meshes of 100K, 500K, 1M, 2M, 4M, up to 20M nodes. The model represents 6 nested cylinders held together with surface to surface contact, meshed with single elastic material solid elements. A prescribed motion on the top and a load on the bottom are imposed for one nonlinear implicit time step with two factors, two solves and four force computations. Figure 1 illustrates a 921,600 solid elements, 1,014,751 nodes problem leading to a 3,034,944 order linear algebra system.

Figure 1: Refs [1] and [2] Cylinder Solid Element Problem Series, 1M nodes

2 Benchmark Systems

Various systems comprised in SGI product line and available through SGI Cyclone™ , HPC on-demand Cloud Computing (see 2.7) were used to run the benchmark described in section 1.

2.1 SGI Octane III

Moderately scalable deskside multi-node system with GigE or Infiniband interconnect in two different configurations:

2.1.1 SGI Octane III Xeon ® X5570

- Dual-socket nodes of 2.93GHz quad-core Xeon X5570, 8 MB cache
- Total Mem: 24 GB Speed: 1333 MHz (0.8 ns)
- GigE and InfiniBand interconnects
- SUSE ® Linux ® Enterprise Server 10 SP2, SGI ProPack™ 6SP3

2.1.2 SGI Octane III Xeon E5540

- Dual-socket nodes of 2.53GHz quad-core Xeon E5540, 8 MB cache
- Total Mem: 24 GB Speed: 1066 MHz (0.9 ns)
- GigE and InfiniBand interconnects
- SLES10SP2 OS, SGI ProPack 6SP3

2.2 SGI Altix XE 1300 cluster

Highly scalable rack-mounted multi-node system with GigE and/or Infiniband interconnects.

- SGI XE270 Administrative/NFS Server node
- SGI XE340 Dual-socket compute nodes of 2.93GHz quad core Xeon X5570 8192KB Cache
- 24GB 1333MHz RAM
- SUSE Linux Enterprise Server 11 SP2, SGI ProPack 6SP3
- SGI Foundation Software 1SP5
- Infiniband ConnectX QDR PCIe Host Card Adapters
- Integrated GigE dual port Network Interface Cards

SGI Altix XE 1300 cluster with dual Ethernet and Infiniband switch is illustrated in Figure 2.

2.3 SGI Altix ICE 8200 cluster

Highly scalable, diskless, integrated cable-free infiniband interconnect rack mounted multi-node system.

- SGI Altix ICE 8200EX DDR SLES11 PP6SP6 Tempo v1.10 Dual-socket compute nodes of 2.93GHz quad core Xeon X5570 24GB RAM in 1066MHz DIMMs
- SGI Altix ICE 8200EX DDR SLES10SP2 PP6SP5 Tempo v1.9 Dual-socket compute nodes of 2.93GHz quad core Xeon X5570 24GB RAM in 1333MHz DIMMs

2.4 SGI Altix 450 and 4700 SMP

Highly scalable Shared Memory Parallel system.

- SGI Altix 4700 128 1.669GHz dual core Itanium ® 9150M 24MB Cache processors, 512GB RAM NUMAlink ® 4

2.5 SGI Altix UV 100, UV 1000 SMP

Highly scalable latest generation Shared Memory Parallel system.

- 12 2.66Ghz 6-core Xeon X7542 (72 cores)
- 192GB RAM
- NUMAlink 5
- SGI Foundation Software, SGI ProPack 7

2.6 Filesystems

Various filesystems may co-exist to use for scratch space:

- Memory-based filesystem /dev/shm
- Root drive
- DAS (Direct Attached Storage)
- NAS (Network Attached Storage)

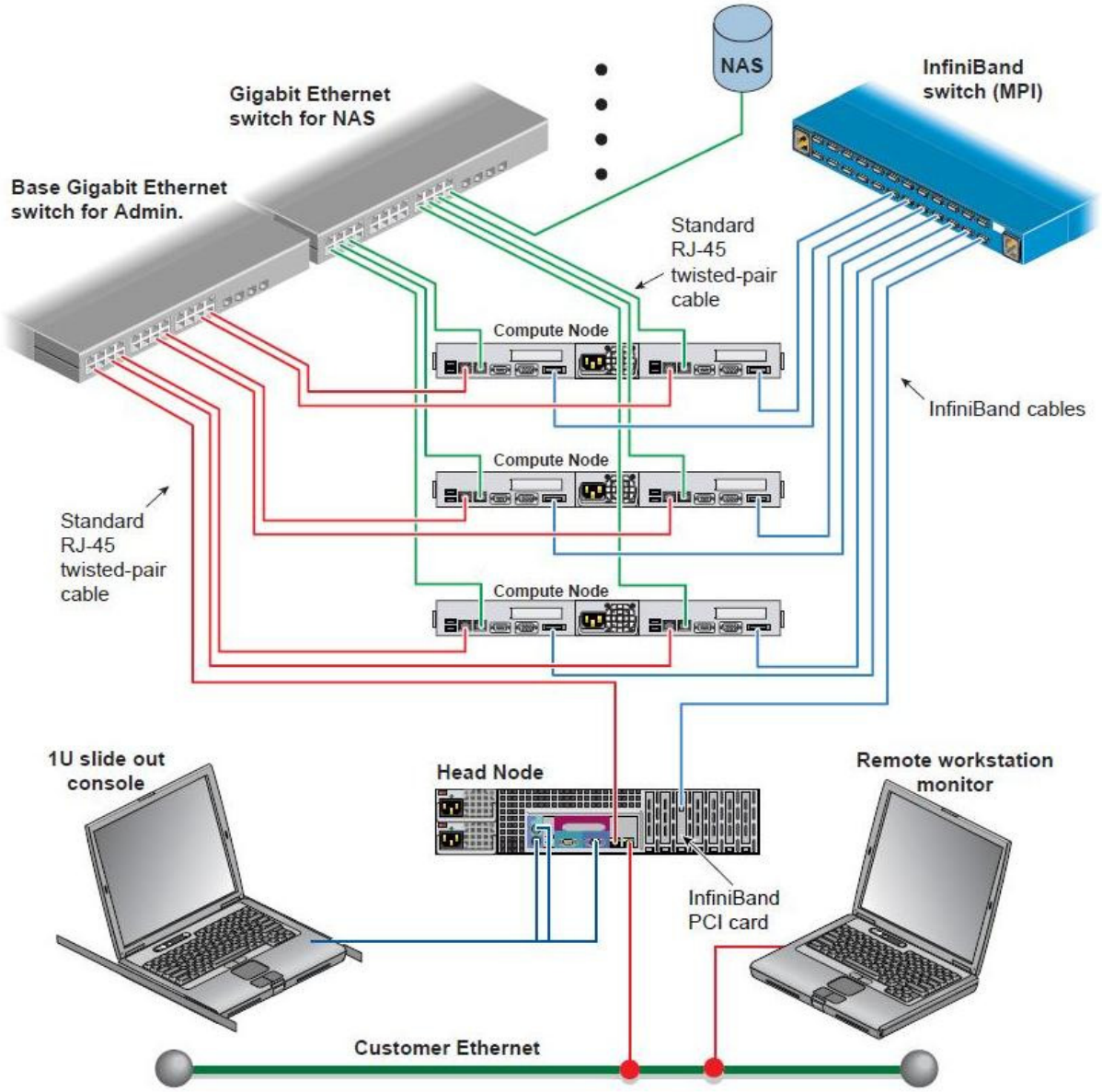

Figure 2: Dual Ethernet and Infiniband switch cluster configuration example

2.7 Cyclone – HPC on-demand Cloud Computing

SGI offers Cyclone, HPC on-demand computing resources of all SGI advanced architectures aforementioned. There are two service models in Cyclone. Software as a Service (SaaS) and Infrastructure as a Service (IaaS). With SaaS, Cyclone customers can significantly reduce time to results by accessing leading-edge open source applications and best-of- breed commercial software platforms from top Independent Software Vendors (ISVs) such as LS-DYNA from LSTC.

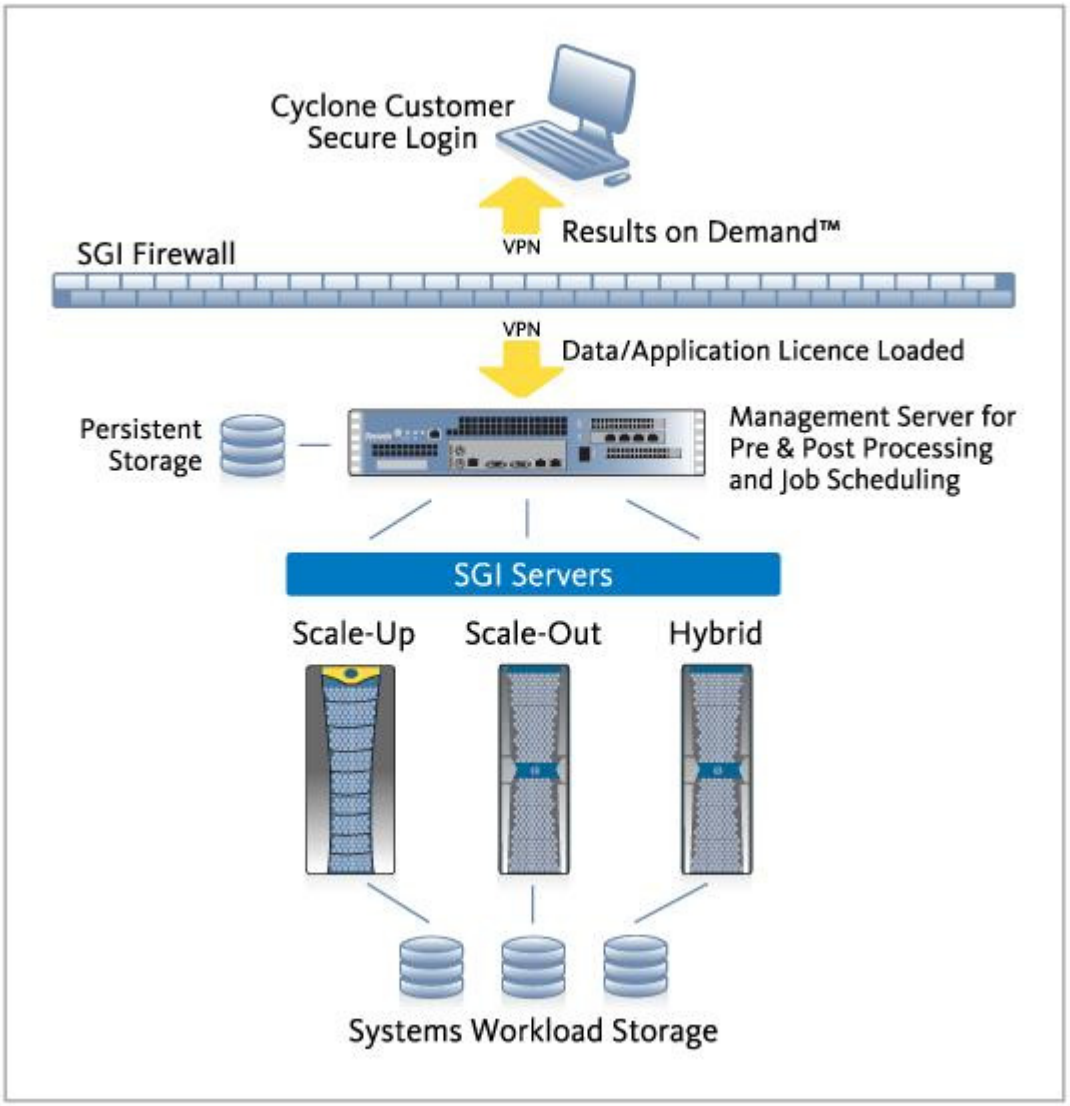

Figure 3: SGI Cyclone – HPC on-demand Cloud Computing

3 LS-DYNA

3.1 Version Used

LS-DYNA/MPP Version 971 Revision 4.2.1 hybrid for SGI Message Passing Toolkit (MPT).

3.2 Parallel Processing Capabilities of LS-DYNA

LS-DYNA exploits parallelism in two paradigms:

- Distributed Memory Parallel (DMP) using MPI Application Programming Interface by reducing the size (while increasing the number) of geometric partitions, decreasing the processing resource requirements for each partition, hence increase efficiency, minimize the size of the common boundary to decrease inter-process communication
- Shared Memory Parallel (SMP) using OpenMP Application Programming Interface.

These two paradigms may simultaneously map themselves on two different system hardware levels:

- Inter-node or cluster parallelism (memory local to each node)–DMP only.
- Intra-node or multi-core parallelism (memory shared by all cores of each node)

The hybrid approach provides increased performance yields and additional possibilities of memory utilization by running SMP on intra-node network in combination with DMP on inter-node and/or intra-node network.

3.3 Submittal procedure

Batch schedulers/resource managers dispatch jobs from a front-end login node to be executed on one or more compute nodes. To achieve the best runtime in a batch environment disk access to input and output files should be placed on the high performance filesystem closest to the compute node. The high performance filesystem could be in-memory filesyste computing environments in-memory filesystem or network attached storage are of course the only options.

Following is the synoptic of a job submission script.

1. Change directory to the local scratch directory on the first compute node allocated by the batch scheduler
2. Copy all input files over to this directory
3. Create parallel local scratch directories on the other compute nodes allocated by the batch scheduler
4. Launch LS-DYNA on the first compute node. The LS-DYNA may itself carry out propagation and collection of various files between launch and the other nodes at start and end of the main analysis execution.

The following keywords are used for the LS-DYNA execution command:

- **-np**: Total number of MPI processes used in a Distributed Memory Parallel job.
- **ncpu**: number of SMP OpenMP threads
- **memory**: Size in words of allocated RAM for each MPI process. A word will be 4 and 8 bytes long for single or double precision executables, respectively

4 Analysis of Benchmark Results

4.1 Effect of Interconnect

The choice of GigE or Infiniband interconnect integrated to a server may affect performance. Figure 4 plots the ratios of elapsed times using GigE over those using Infiniband interconnect. For the benchmark series used in this study, factors above 1.2 can be observed for all sizes of datasets and at higher MPI task counts, the performance difference can be over 60%.

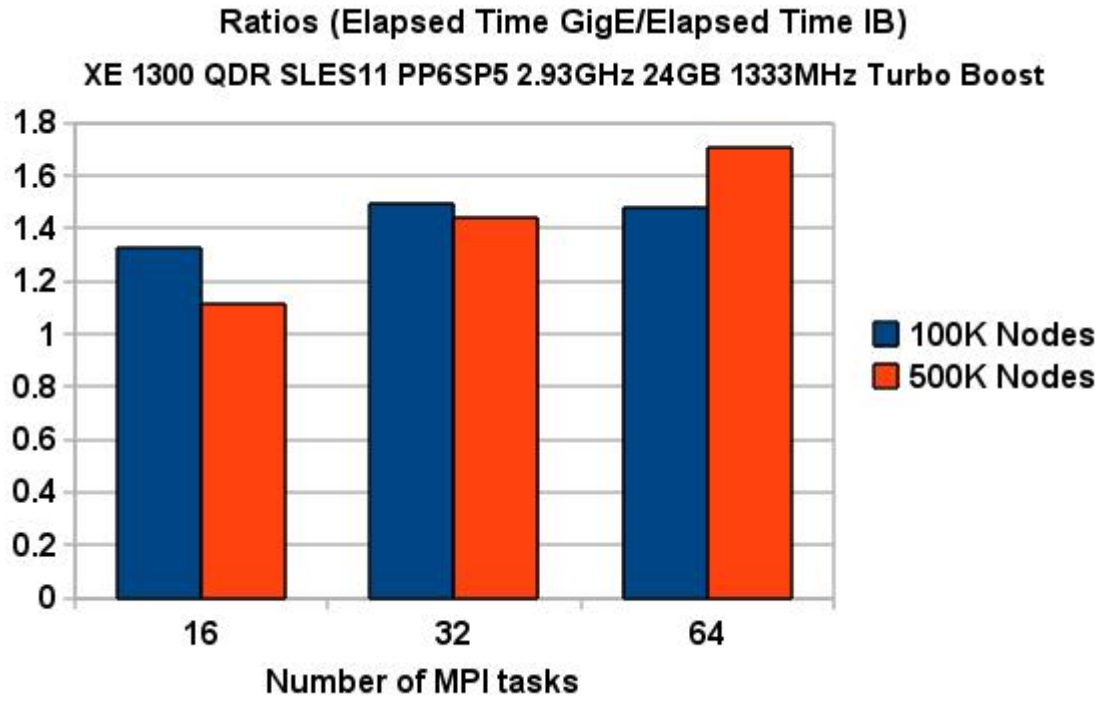

Figure 4: Effect of Interconnect for 100K and 500K Nodes datasets and increasing numberof MPI tasks

4.2 Effect of core frequency

The processor core frequency can be expected to affect performance in a non-linear fashion since the problem and the system may be subject to other bottlenecks than pure arithmetic processing. Figure 5 plots the elapsed times normalized by the slowest processor. For the benchmark series used in this study, increased core frequencies improve performance linearly for 100K Nodes/1 and 8 core runs but show diminishing returns in the other more complex cases. The ideal performance is represented by the straight line 'Frequencies'.

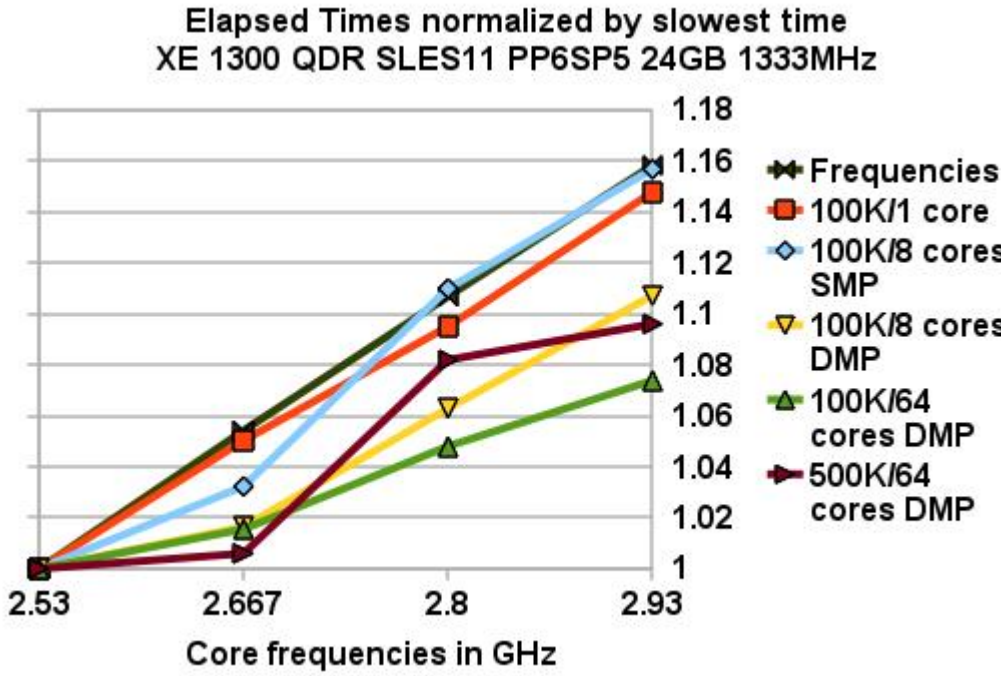

Figure 5: Effect of core frequency on performance for various test cases, SGI XE1300 Clusterbased on Intel Xeon 5500 Series Processors (Turbo Boost disabled)

4.3 Effect of Turbo Boost

The Turbo Boost is a feature first introduced in the Xeon 5500 series processors, which can increase performance by increasing the core operating frequency within the controlled limits of the processor. For most simulations, utilizing Turbo Boost technology can result in improved runtimes. By design the mode of activation is a function of how many cores are active at a given moment as maybe the case when OpenMP threads or MPI processes are idle under their running parent. Similarly to 4.2, its effects may be mitigated by the presence of other performance bottlenecks than pure arithmetic processing. Figure 6 plots the ratios between elapsed times with Turbo Boost OFF versus ON. It shows that for the benchmark series used in this study, Turbo Boost improves performance for 100K Nodes/1 core run up to the ratio of the maximum frequency over nominal value which is 13% (magnitude represented by first bar) but to a lesser extent in the other cases.

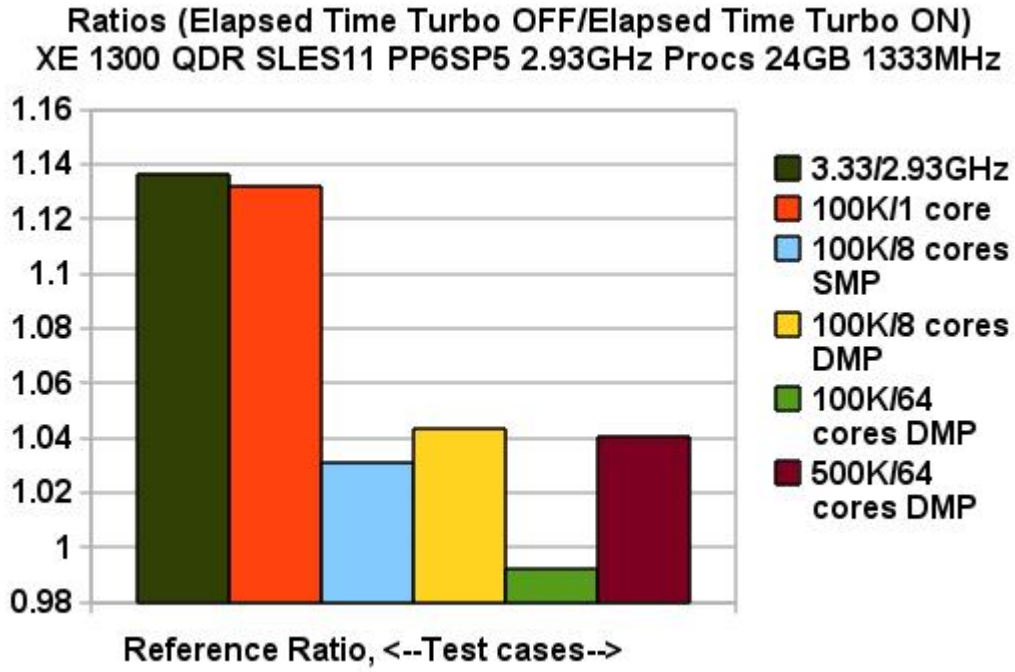

Figure 6: Effect of Turbo Boost on performance on various test cases

4.4 Effect of Hyper-Threading Technology

Hyper-Threading (HT) is a feature which can increase performance for multi-threaded or multi-process applications. It allows a user to run 16 (instead of 8) OpenMP threads or MPI processes on 8 physical cores per node. It is sometimes beneficial for 1 or 2 nodes but at and above 3 nodes communication costs added by the doubling of threads or MPI processes mitigates the benefits. Figure 7 plots the ratios between elapsed times without using Hyper- Threading vs using it. Based on the benchmark study no improvement is observed for 100K and 500K Nodes.

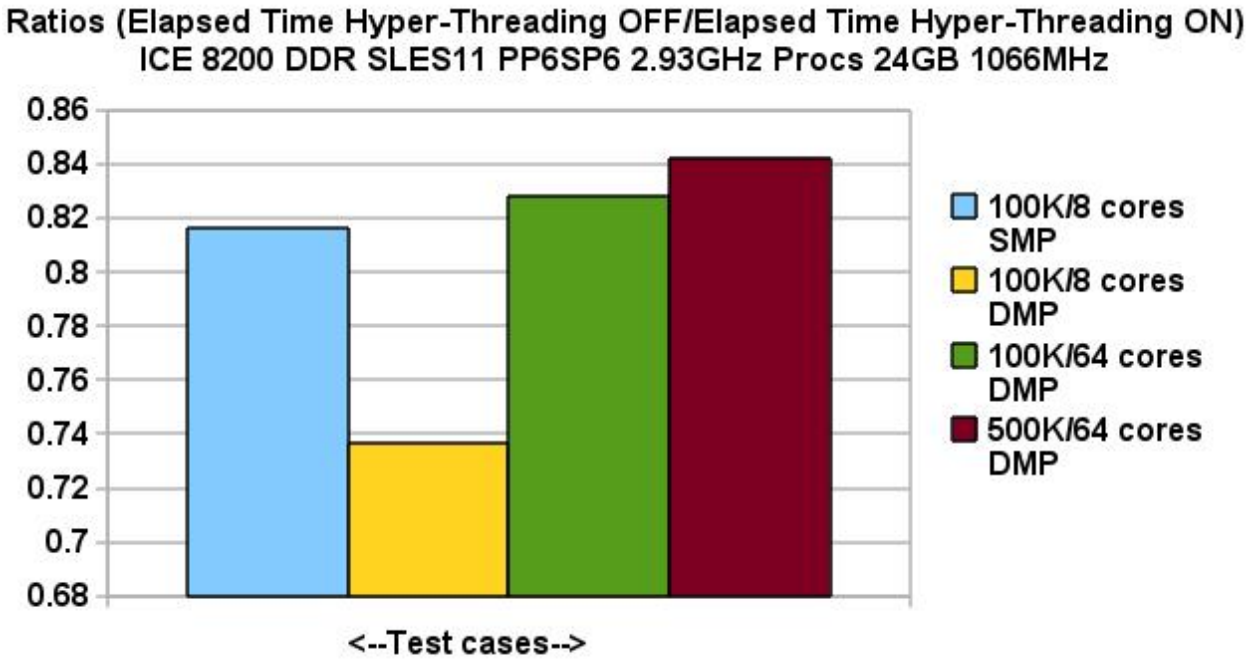

Figure 7: Negative effect of Hyper-Threading on test cases tried

4.5 Effect of Memory Speed

Memory speed may be important when data motion represents either a bandwidth bottleneck or latency limitation. Figure 8 plots the ratios between elapsed times using 1066MHz vs 1333MHz DIMMs. It shows that for the benchmark series used in this study, in the two cases run, the improvements are far below the nominal ratio of memory speed which is 25% (magnitude represented by first bar).

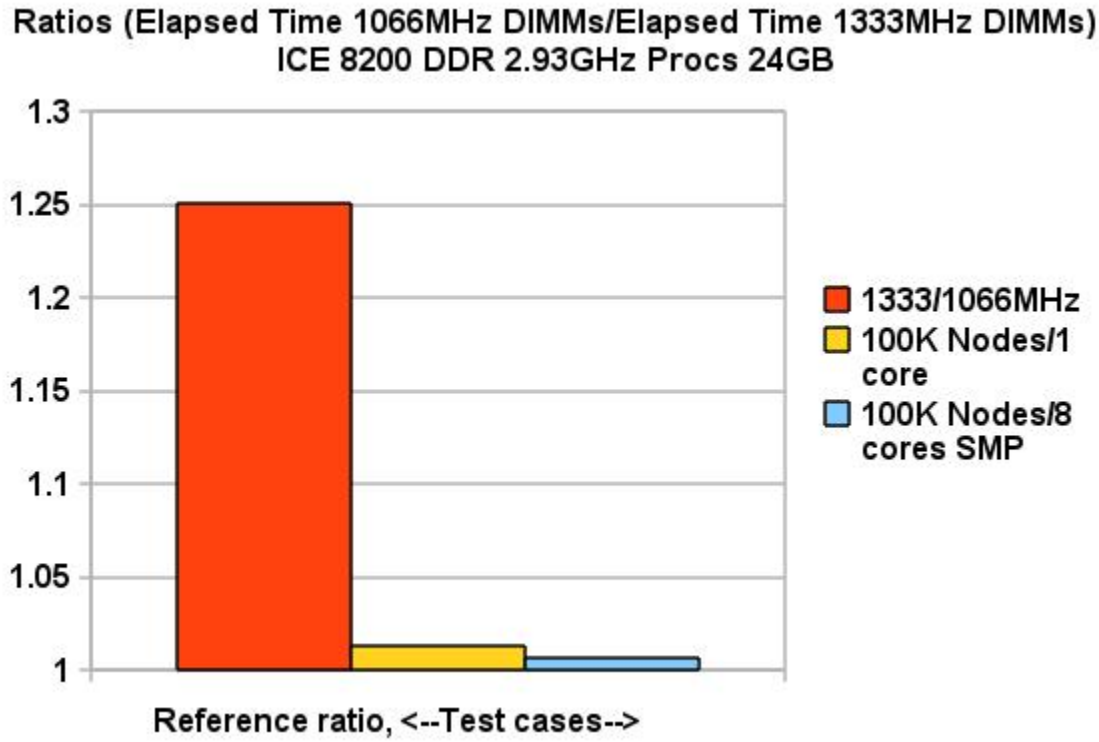

Figure 8: Effect of memory speed is negligible for test cases tried

4.6 Effect of File System

Figure 9 plots the ratios between elapsed times using disk vs memory file system. It shows that for the benchmark series used in this study, using /dev/shm when possible can save up to 10% elapsed time.

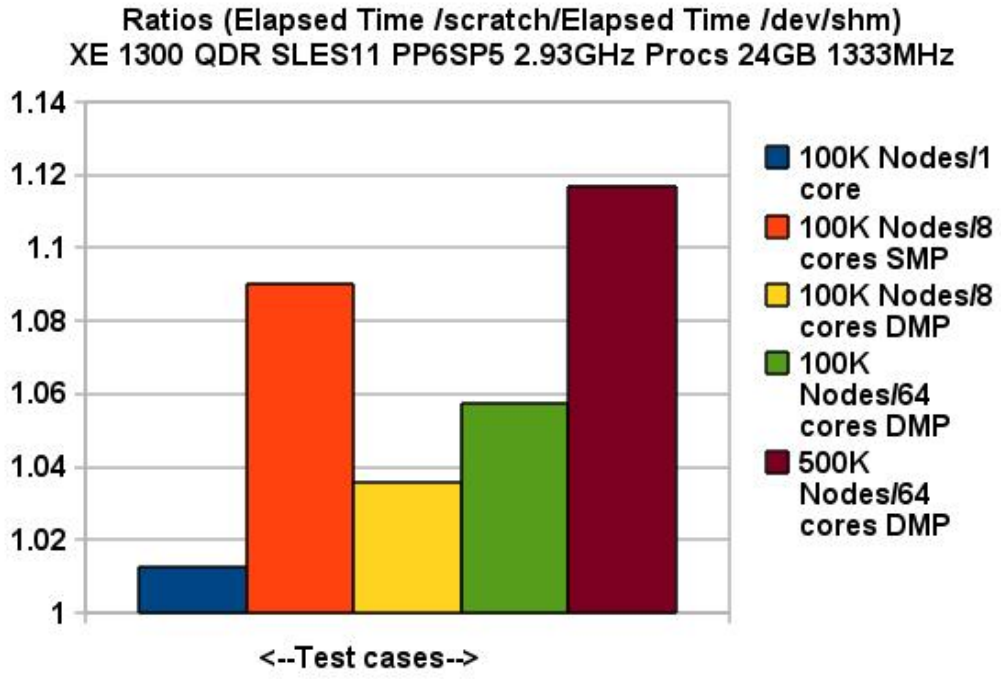

Figure 9: Effect of filesystem

4.7 Effect of MPI tasks and OpenMP thread allocation across nodes and cores

4.7.1 Optimizing turnaround time

Table 1 shows how different combinations of number of MPI tasks (np) and OpenMP threads (ncpu) can be mapped across a given number of nodes and a total number of cores affecting elapsed time (last column). Up to 512 cores, (column 3), were used for these experiments on the SGI Altix ICE 8200 DDR SLES10SP2 PP6SP5 Tempo v1.9 2.93GHz Procs 24GB 1333MHz. All 8 physical cores within each node are used by either an MPI task or an OpenMP thread except in the first three lines (single node runs). The fastest elapsed times are obtained with a mix of MPI tasks and OpenMP threads which shows that hybrid mode is an improvement over either pure SMP (OpenMP threads) or DMP (MPI tasks) available previously. The 'memory' keyword used for a MPI task to process factorization and solution of the partitioned linear system in-core is shown in column 6. The next column multiplies this amount by the number of tasks running within each node to give the memory used therein. For the 100K Nodes dataset used, it is always within the 24GB capacity of the node. The next column is the total memory used across all nodes giving an indication of how it would fit within a Shared Memory Parallel system such as SGI Altix 450, 4700 or SGI Altix UV 100, 1000. Looking at this table as a whole allows one to look at tradeoffs in applying various number of nodes to problems for maximizing throughput or turnaround times.

Dataset	Nodes	Cores	np	ncpu	memory	mem/node	mem total	Seconds
100K	1	8	1	1	4032	4032	4032	757
100K	1	8	1	2	4032	4032	4032	405
100K	1	8	1	4	4032	4032	4032	237
100K	1	8	1	8	4032	4032	4032	152
100K	1	8	1	16	4032	4032	4032	184
100K	8	64	64	1	112	896	7168	32
100K	8	64	32	2	240	960	7680	30
100K	8	64	16	4	352	704	5632	31
100K	8	64	8	8	576	576	4608	35
100K	16	128	128	1	64	512	8192	24
100K	16	128	64	2	112	448	7168	23
100K	32	256	256	1	40	320	10240	27
100K	32	256	128	2	56	224	7168	21
100K	32	256	64	4	112	224	7168	18
100K	32	256	32	8	240	240	7680	20
100K	64	512	256	2	40	160	10240	24
100K	64	512	128	4	64	128	8192	21
100K	64	512	64	8	120	120	7680	19

Table 1: Hybrid mode MPI tasks and Threads for 100K Nodes dataset on SGI Altix ICE8200, (memory is in MB)

4.7.2 Optimizing memory footprint

The larger 500K Nodes case of table 2 shows how 32 MPI tasks produce a memory used per node close to what might be available in some installations (16GB nodes for example). In that case, increasing the number of MPI tasks to 64 would allow the solution to run in-core. An alternative is to use a Shared Memory Parallel architecture as shown in the last 4 lines run on an SGI Altix UV 100 with 12 2.66Ghz 6-core Xeon X7542 (72 cores) with a total of 192GB as a single node.

Dataset	Nodes	Cores	np	ncpu	memory	mem/node	mem total	Seconds
500K	4	32	32	1	1872	14976	59904	967
500K	8	64	64	1	960	7680	61440	591
500K	9	72	72	1	1024	8192	73728	570
500K	9	72	36	2	1240	4960	44640	577
500K	9	72	24	3	2240	4480	40320	633
500K	9	72	72	1	880	63360	63360	608
500K	9	72	36	2	1372	49392	49392	577
500K	9	72	24	3	2536	60864	60864	613
500K	9	72	18	4	2824	50832	50832	584

Table 2: Hybrid mode MPI tasks and Threads for 500K Nodes dataset on SGI Altix XE1300 and SGI Altix UV 100 (memory is in MB)

Conclusions

This study showed how interconnect, core frequency, Turbo Boost, Hyper-Threading, memory speed, file system effects on implicit LS-DYNA performance can be gauged for various sizes of datasets. At the same time, using the right combination of LS-DYNA hybrid mode parallelism features, most efficient use of either shared or distributed memory systems can be achieved to optimize throughput or turnaround times. Both these insights and optimizations can be used to architect a solution using the full range of systems from SGI to meet complex analysis requirements.

Attributions

References

[1] Dr. C. Cleve Ashcraft, Roger G. Grimes, and Dr. Robert F. Lucas. “A Study of LS-DYNA Implicit Performance in MPP”. In Proceedings of 7th European LS-DYNA Conference, Austria, 2009.
[2] Dr. C. Cleve Ashcraft, Roger G. Grimes, and Dr. Robert F. Lucas. “A Study of LS-DYNA Implicit Performance in MPP (Update)”. 2009.
[3] SGI. SGI Developer’s Guide. Silicon Graphics International, Fremont, California, 2009.