Matching LS-DYNA Explicit, Implicit, Hybrid technologies with SGI architectures

Olivier Schreiber*, Tony DeVarco*, Scott Shaw* and Suri Bala†

**SGI, †LSTC*

Abstract

LSTC has now integrated Explicit, Implicit solver technologies into a single hybrid code base allowing seamless switching from large time steps transient dynamics to linear statics and normal modes analysis. There are multiple computer architectures available from SGI to run LS-DYNA. They can all run LSTC solvers using Shared Memory Parallelism (SMP), Distributed Memory Parallelism (DMP) and their combination (Hybrid Mode) as supported by LS-DYNA. Because computer resources requirements are different for Explicit and Implicit solvers, this paper will study how advanced SGI computer systems, ranging from multi-node Distributed Memory Processor clusters to Shared Memory Processor servers address the computer resources required and what tradeoffs are involved. The paper will also outline the specifications of running LS-DYNA jobs on Cyclone, SGI's HPC cloud computing infrastructure using d3View. d3View is a simulation data management and visualization software that extends the use of HPC by performing simulation data extraction and analysis on the compute nodes.

Introduction

The subject of this paper is to evaluate the use of SGI® ICE and SGI ®UV architectures using the most recent technologies for Shared Memory Parallel (SMP), Distributed Memory 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 Reference [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 Reference [3].

1 Benchmark Systems

Various systems comprised in SGI product line and available through SGI Cyclone™ , HPC on-demand Cloud Computing were used to run the benchmarks.

1.1 SGI ICE cluster

Highly scalable, diskless, integrated cable-free infiniband interconnect rack mounted multi-node system the SGI® ICE integrated blade cluster was designed for today's data intensive problems. This innovative new platform from SGI raises the efficiency bar, easily scaling to meet virtually any processing requirements without compromising ease of use, manageability or price/performance. SGI Altix ICE delivers unsurpassed customer value, with breakthrough density, efficiency, reliability and manageability (Figure 1).

- Intel Xeon 5500 2.93GHz quad-core or 5600 3.46GHz six-core
- Two single-port ConnectX-2 IB HCA
- 12 DDR3 1066 MHz or 1333 MHz ECC DIMM slots per blade
- SGI Tempo management tools™

- SUSE Linux Enterprise Server 11 SP2, SGI ProPack 6SP3 for Linux® and Altair® PBSpro workload manager.

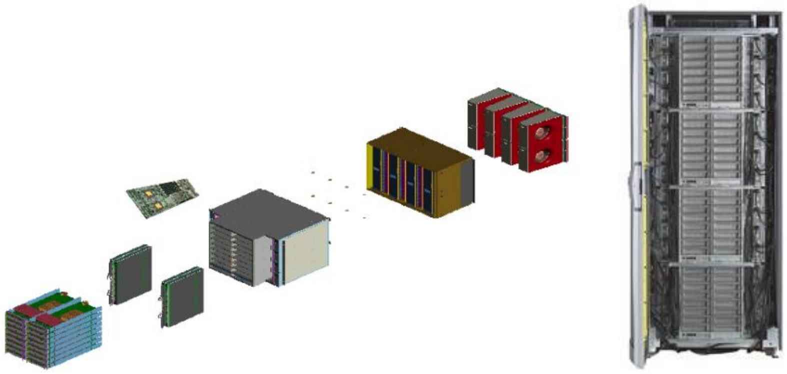


Figure 1: SGI Altix ICE cluster and IRU

1.2 SGI® UV 10, UV 100, UV 1000 (SMP)

Altix® UV scales to extraordinary levels-up to 256 sockets (2,048 cores, 4096 threads) with architectural support to 262,144 cores (32,768 sockets). Support for up to 16TB of global shared memory in a single system image, enables Altix® UV to remain highly efficient at scale for applications ranging from in-memory databases, to a diverse set of data and compute-intensive HPC applications. With this platform, it is simpler for the user to access huge resources for programming via a familiar OS, without the need for rewriting their software to include complex communication algorithms. (Figure 2)

Figure 2: SGI UV 10, UV 100, UV 1000 SMP

- 6-core Intel Xeon 7542 2.66GHz
- NUMAlink R 5
- SUSE Linux Enterprise Server 11 SP2, SGI ProPack 6SP3 for Linux®

1.3 Access to benchmark systems

SGI offers Cyclone, HPC on-demand computing resources of all SGI advanced architectures aforementioned (Figure 3). There are two service models in Cyclone. Software as a Service

(SaaS) and Infrastructure as a Service (IaaS) (Figure 4). With SaaS, Cyclone customers can reduce time to results by accessing leading-edge open source applications and best-of- breed commercial software platforms from top Independent Software Vendors (ISV's) like LSTC.

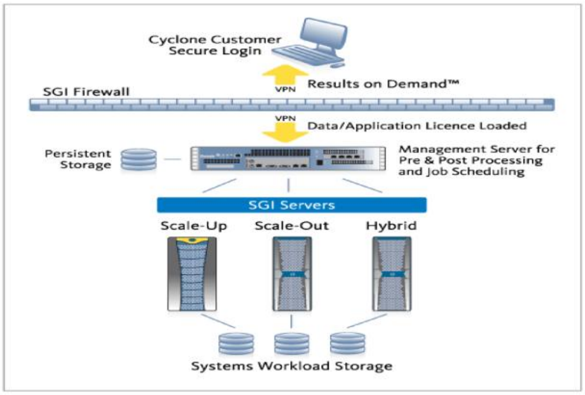


Figure 3: SGI Cyclone – HPC on-demand Cloud Computing

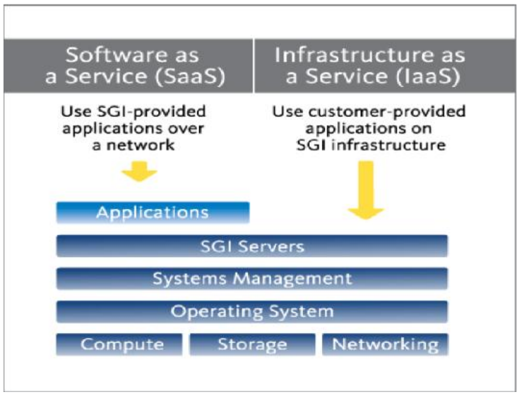


Figure 4: SGI Cyclone Services

The physical elements of Cyclone feature:

- Pre-configured, pre-certified applications and tools
- High speed Scale-Up and Scale-Out platforms
- High speed processors
- High speed networking (NUMAlink, InfiniBand)
- Non-virtualized environments
- Dedicated management node for security
- SSH or d3View web portal access
- From scratch storage to long-term storage
- Data exchange service

1.4 d3View

d3VIEW is a web based software that provides users with a single unified interface for submitting, monitoring and visualizing LS-DYNA simulation results. Coupled with its advanced visualization features and multiple-simulation comparison capabilities, d3VIEW is the industry leader in providing a platform for simulation engineers in the area of simulation data visualization and collaboration.

d3VIEW has been integrated with SGI Cyclone clusters to provide users an instant access for running complex simulations. Jobs can be submitted and monitored from any internet-enabled device. d3VIEW also provides a "Job Preview" function that allows users to get quick peek at the ongoing simulations in real-time. Users can also send signals to LS-DYNA or alter job properties while the job is running on Cyclone.

Once the job completes, d3VIEW processes the results which otherwise is done manually to present the user an "overview" of the simulation that emphases simulation quality and structural performance. Depending on the result overview, users can then make quick "size" changes and resubmit the job or download the data set to perform additional calculations.

2 LS-DYNA

2.1 Version Used

LS-DYNA/MPP ls971 R5.1.1 hybrid for Message Passing Interface R3.2.1 is faster than R5.1.1 by 25% (neon) to 35% (car2car) because at R4.2.1, coordinate array went to double precision for slow motion simulation.

2.2 Parallel Processing Capabilities of LS-DYNA

2.2.1 Nomenclature

A node is synonymous to one host or one blade or one chassis, identified by one MAC address and one IP address. It comprises two sockets (most common) or more on which are plugged in a processor with four (quad-core), six (hexa-core), eight or twelve cores on each.

2.2.2 Background

Parallelism in scientific/technical computing exist in two paradigms implemented separately or recently combined in the same so-called Hybrid code:

- **Shared Memory Parallelism** (SMP) appeared in the ’80s around DO loop processing or subroutine spawning and consolidated on the OpenMP (Open Multi-Processing) Application Programming Interface and Pthreads standard. Parallel efficiency is affected by the ratio of arithmetic operations versus data access or DO loop granularity.
- **Distributed Memory Parallelism** (DMP) appeared in the late ’90’s around physical or mathematical domain decomposition and consolidated on the MPI Application Programming Interface. Parallel efficiency is affected by the boundaries created by the partitioning.

These two paradigms can map themselves on two different system hardware levels:

- Shared Memory systems or single nodes with memory shared by all cores.
- Cluster Nodes with their own local memory, i.e. Distributed Memory systems.

Shared Memory Parallelism cannot span cluster nodes either communication or memory-wise. On the other hand, Distributed Memory Parallelism can be used within a Shared Memory system. Since DMP is of a coarser granularity than SMP, it is preferable, when possible to run DMP within Shared Memory Systems.

3 Benchmarks Description

The benchmarks used are the three TopCrunch (http:www.topcrunch.org) benchmark datasets.

3.1 Neon Refined Revised

The benchmark consists of a frontal crash with initial speed at 31.5 miles/hour with a total model size of 535k elements, 532,077 shell elements, 73 beam elements, 2,920 solid elements, 2 contact interfaces, 324 materials, and a simulation time of 30 ms (29,977 cycles) Figure 5). The vehicle model was created by National Crash Analysis Center (NCAC) at George Washington University (Publicly available vehicle crash analysis model based on 1996 Plymouth Neon). The dataset file causes LS-DYNA to write 68,493,312 Bytes d3plot and 50,933,760 Bytes d3plot01 files at 8 time steps from start to end point (114MB).

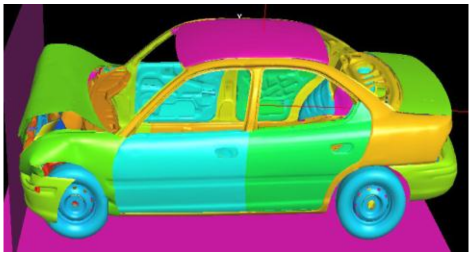


Figure 5: Neon Refined Revised

3.2 3 Vehicle Collision

The benchmark consists of a van crashing into the rear of a compact car, which, in turn, crashes into a midsize car with a total model size of 794,780 elements, 785,022 shell elements, 116 beam elements, 9,642 solid elements, 6 contact interfaces, 1,052 materials, and a simulation time of 150 ms (149,881 cycles) (Figure 6). The vehicle models created by National Crash Analysis Center (NCAC) at George Washington University, and assembled into the input file by Mike Berger, consultant to LSTC. The dataset file causes LS-DYNA to write 65,853,440 Bytes d3plot and 33,341,440 Bytes d3plot[01-19] files at 20 time steps from start to end point (667MB).

According to LSTC, the 3cars model is very difficult to scale well: most of the contact work is in two specific areas of the model, and it is hard, if not impossible, to evenly spread that work out across a large number of processes. Particularly as the”active” part of the contact (which part is crushing the most) changes with time, so the computational load of each process will change with time.

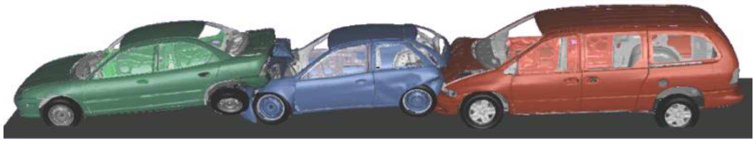


Figure 6: Vehicle Collision

3.3 car2car

The benchmark consists of an angled 2 vehicle collision (figure 7). The vehicle models are based on NCAC minivan model, created by Dr. Makino and Supplied by Dr. Tsay, LSTC, on Feb. 13, 2006. The termination time was modified per John Hallquist to .120 on March 7, 2006. The dataset causes LS-DYNA to write 201,854,976 Bytes d3plot and 101,996,544 Bytes d3plot[01-25] files at 26 time steps from start to end point (2624MB).

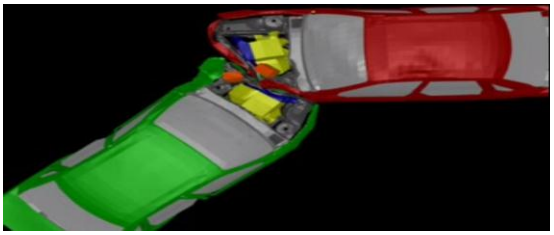


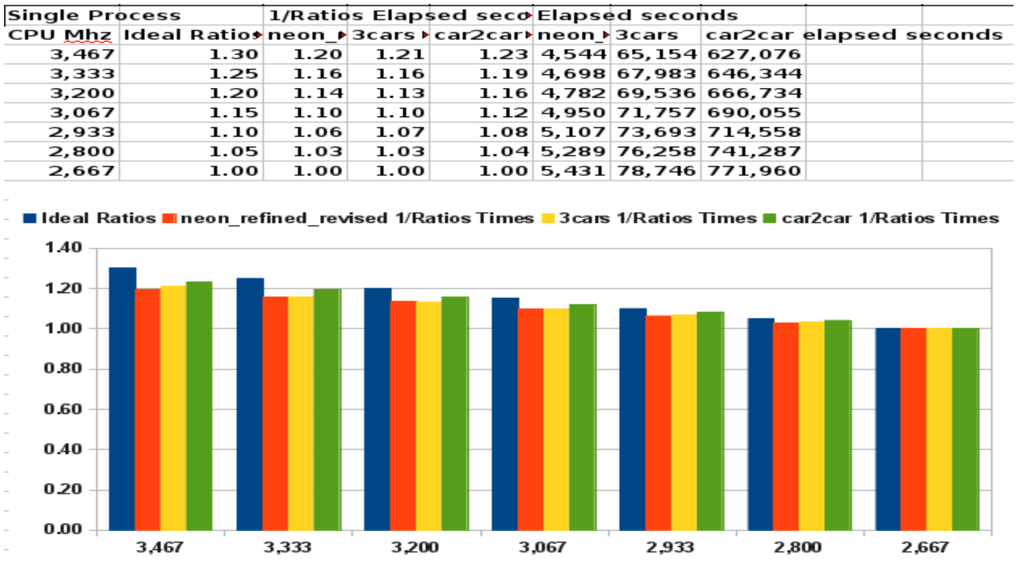
Figure 7: car2car

4 Effect of processor frequency

Elapsed time is not inversely proportional to CPU frequency as Figure 8 illustrates, for a serial run. LS-DYNA's explicit computations are BLAS 1, i.e. vector-vector dominated. In a serial run, similarly to STREAM benchmark, there is one core processing data at a rate limited by the bandwidth of the channel of the first memory subsystem (cache) with a slower frequency than the core frequency, causing performance to be less than proportional to core frequency.

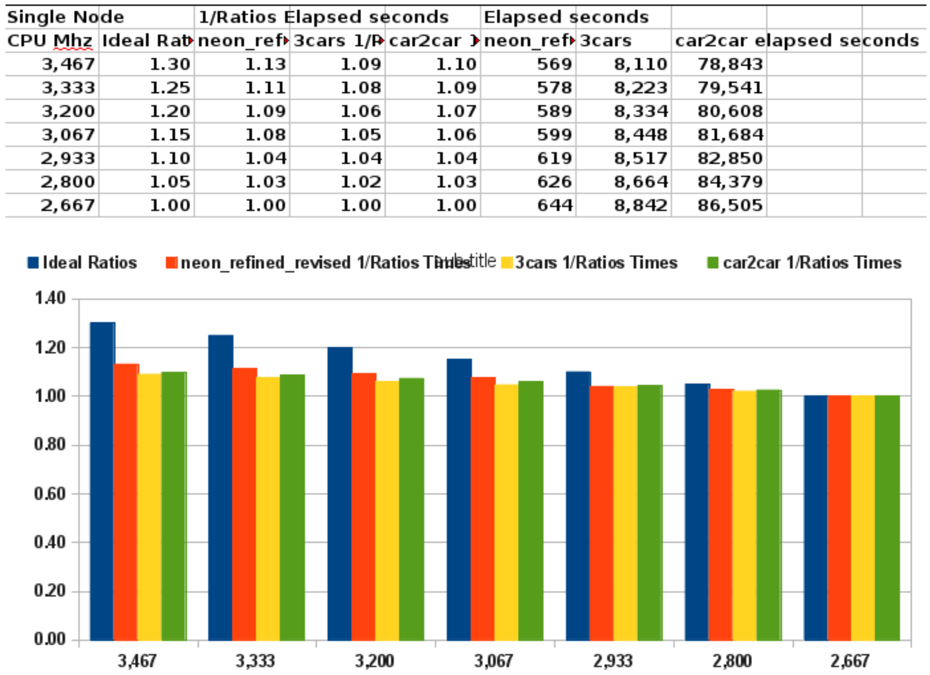
The fully subscribed case of 12 MPI processes on the 12 physical cores available (Figure 9) causes the performance increase due to CPU frequency to be even less than for the serial run because the memory bandwidth becomes another limiting factor. Specifically, aggregate (full-node) bandwidth is not necessarily equal to the serial bandwidth multiplied by the number of physical cores.

When 8 nodes are used (Figure 10) sensitivity to CPU frequency is higher because although all cores are utilized, bandwidth requirements are relieved by the decomposition into smaller domains so each core needs to stream less data. By the same rationale neon_refined_revised is more sensitive to CPU frequency than the more bandwidth-sensitive 3cars and car2car datasets.



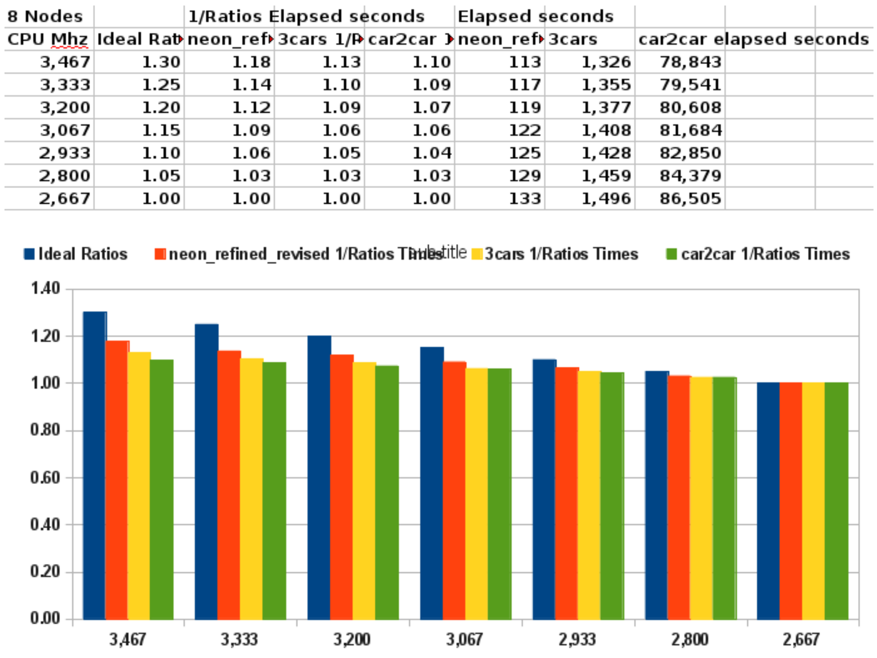
Single Process		1/Ratios Elapsed seco			Elapsed seconds		
CPU Mhz	Ideal Ratio	neon_	3cars	car2car	neon_	3cars	car2car elapsed seconds
3,467	1.30	1.20	1.21	1.23	4,544	65,154	627,076
3,333	1.25	1.16	1.16	1.19	4,698	67,983	646,344
3,200	1.20	1.14	1.13	1.16	4,782	69,536	666,734
3,067	1.15	1.10	1.10	1.12	4,950	71,757	690,055
2,933	1.10	1.06	1.07	1.08	5,107	73,693	714,558
2,800	1.05	1.03	1.03	1.04	5,289	76,258	741,287
2,667	1.00	1.00	1.00	1.00	5,431	78,746	771,960

Figure 8: Ideal performance gains versus actual for single process LS-DYNA explicit runs



Single Node		1/Ratios Elapsed seconds			Elapsed seconds		
CPU Mhz	Ideal Rat	neon_ref	3cars 1/R	car2car	neon_ref	3cars	car2car elapsed seconds
3,467	1.30	1.13	1.09	1.10	569	8,110	78,843
3,333	1.25	1.11	1.08	1.09	578	8,223	79,541
3,200	1.20	1.09	1.06	1.07	589	8,334	80,608
3,067	1.15	1.08	1.05	1.06	599	8,448	81,684
2,933	1.10	1.04	1.04	1.04	619	8,517	82,850
2,800	1.05	1.03	1.02	1.03	626	8,664	84,379
2,667	1.00	1.00	1.00	1.00	644	8,842	86,505

Figure 9: Ideal performance gains versus actual for 12 MPI processes LS-DYNA explicit runs



8 Nodes		1/Ratios Elapsed seconds			Elapsed seconds		
CPU Mhz	Ideal Rat	neon_ref	3cars 1/R	car2car	neon_ref	3cars	car2car elapsed seconds
3,467	1.30	1.18	1.13	1.10	113	1,326	78,843
3,333	1.25	1.14	1.10	1.09	117	1,355	79,541
3,200	1.20	1.12	1.09	1.07	119	1,377	80,608
3,067	1.15	1.09	1.06	1.06	122	1,408	81,684
2,933	1.10	1.06	1.05	1.04	125	1,428	82,850
2,800	1.05	1.03	1.03	1.03	129	1,459	84,379
2,667	1.00	1.00	1.00	1.00	133	1,496	86,505

Figure 10: Ideal performance gains versus actual for 96 MPI processes LS-DYNA explicit runs

5 Effect of topology

Interconnect influence on synthetic and application benchmarks have been addressed comprehensively, Figure 11 illustrates that for the 3cars benchmark, the effect of standard, enhanced hypercube, 'all to all', 'fat tree' on one or two planes is below seven percent.

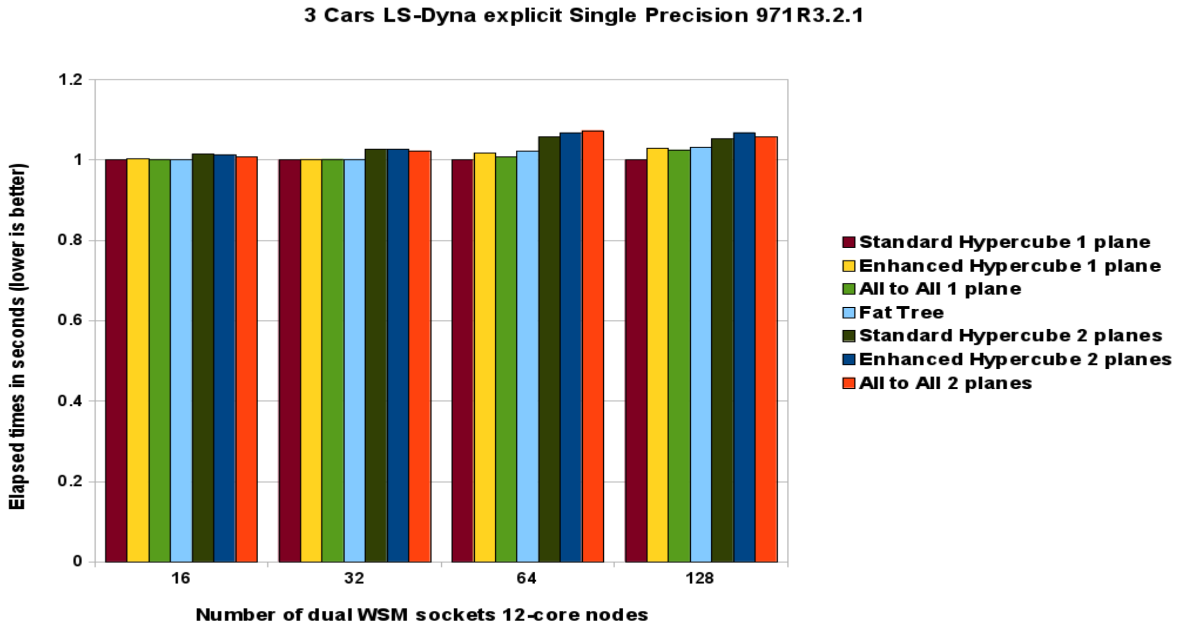


Figure11: Topology LS-DYNA

6 Effect of hybrid parallelism

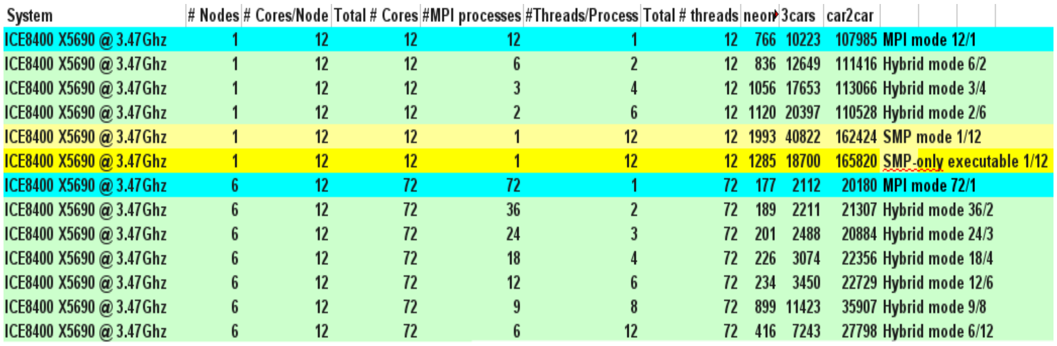
6.1 MPI tasks and OpenMP thread allocation across nodes and cores

For LS-DYNA, the deployment of processes, threads and associated memory is achieved with the following keywords in 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.

An MPI library capability to bind an MPI rank to a processor core is key to control performance because of the multiple node/socket/core environments. From [reference 5], '3.1.2 Computation cost-effects of CPU affinity and core placement [...]HP-MPI currently provides CPU-affinity and core-placement capabilities to bind an MPI rank to a core in the processor from which the MPI rank is issued. Children threads, including SMP threads, can also be bound to a core in the same processor, but not to a different processor; additionally, core placement for SMP threads is by system default and cannot be explicitly controlled by users.[...]'. In contrast, MPT, through the omplace command uniquely provides convenient placement of Hybrid MPI processes/OpenMP threads and Pthreads within each node. This MPI library is linklessly available through the PerfBoost facility bundled with SGI ProPack. PerfBoost provides a Platform-MPI, IntelMPI, OpenMPI, HP-MPI ABI-compatible interface to SGI MPT MPI.

6.2 Comparison MPI only, Hybrid and SMP only



System	# Nodes	# Cores/Node	Total # Cores	#MPI processes	#Threads/Process	Total # threads	neon	3cars	car2car	
ICE8400 X5690 @ 3.47Ghz	1	12	12	12	1	12	766	10223	107985	MPI mode 12/1
ICE8400 X5690 @ 3.47Ghz	1	12	12	6	2	12	836	12649	111416	Hybrid mode 6/2
ICE8400 X5690 @ 3.47Ghz	1	12	12	3	4	12	1056	17653	113066	Hybrid mode 3/4
ICE8400 X5690 @ 3.47Ghz	1	12	12	2	6	12	1120	20397	110528	Hybrid mode 2/6
ICE8400 X5690 @ 3.47Ghz	1	12	12	1	12	12	1993	40822	162424	SMP mode 1/12
ICE8400 X5690 @ 3.47Ghz	1	12	12	1	12	12	1285	18700	165820	SMP-only executable 1/12
ICE8400 X5690 @ 3.47Ghz	6	12	72	72	1	72	177	2112	20180	MPI mode 72/1
ICE8400 X5690 @ 3.47Ghz	6	12	72	36	2	72	189	2211	21307	Hybrid mode 36/2
ICE8400 X5690 @ 3.47Ghz	6	12	72	24	3	72	201	2488	20884	Hybrid mode 24/3
ICE8400 X5690 @ 3.47Ghz	6	12	72	18	4	72	226	3074	22356	Hybrid mode 18/4
ICE8400 X5690 @ 3.47Ghz	6	12	72	12	6	72	234	3450	22729	Hybrid mode 12/6
ICE8400 X5690 @ 3.47Ghz	6	12	72	9	8	72	899	11423	35907	Hybrid mode 9/8
ICE8400 X5690 @ 3.47Ghz	6	12	72	6	12	72	416	7243	27798	Hybrid mode 6/12

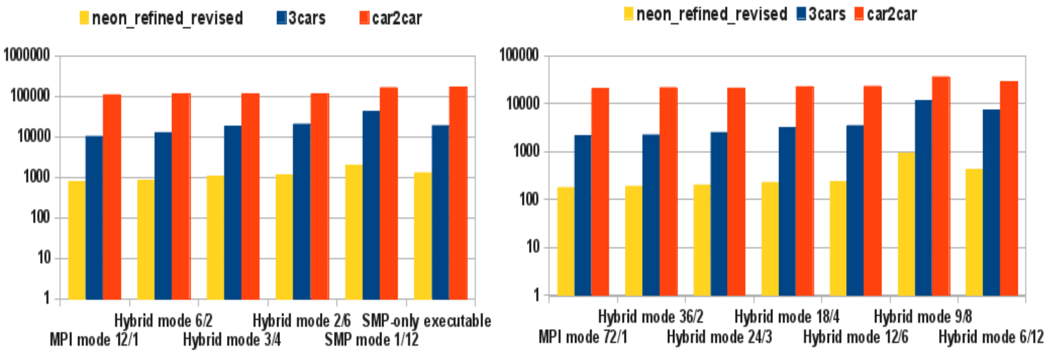


Figure 12: SMP vs. DMP vs. Hybrid

The different cases involving DMP, SMP and Hybrid modes have been organized in the following progression for the case of an ICE 8400 Xeon 5690 cluster single node and six-node system: (SMP term is used for both computation mode and system).

- MPI mode (BLUE). This is the case where only MPI is used on all cores available. The command typically used would look like:
  ```
  mpirun -np Max#Cores lsdynaHybridExecutable inputFile ncpu=1
  ```
- Increasing degrees of Hybrid mode (GREEN). This are the cases where combinations of MPI processes and threads are used. The command typically used would look like:
  ```
  mpirun -np decreasingCores lsdynaHybridExecutable inputFile
  ncpu=increasingCores
  ```

As shown in the Figure 12, the arguments for -np and ncpu are arranged so that the total number of threads fully uses all cores available.

- SMP mode with Hybrid executable (LIGHT YELLOW). This is the case where only thread parallelism is used on all cores available using the same executable. The command typically used would look like:

```
mpirun -np 1 lsdynaHybridExecutable inputFile
ncpu=Max#CoresOn1Node
```

- SMP mode with SMP-only executable (BRIGHT YELLOW). This is the case where only thread parallelism is used on all cores available using the SMP-only executable. The command typically used would look like:
  ```
  lsdynaSMPexecutable myInputFile ncpu=Max#CoresOn1Node
  ```

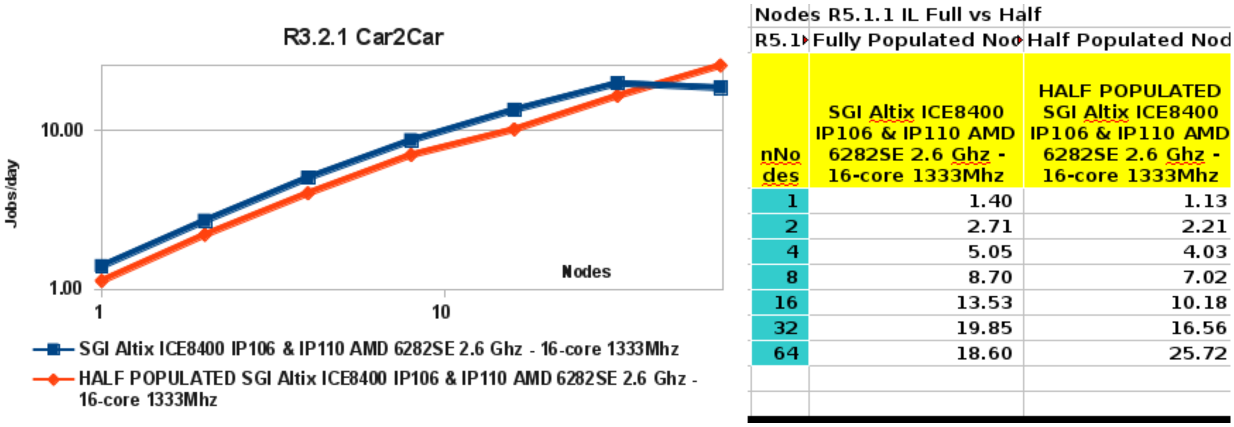
Figure 12 shows Hybrid mode does not bring benefits for LS-DYNA explicit compared to implicit as was reported in [reference 5] and [reference 6]. For this particular hardware, no case shows benefits. Pure MPI is remarkably efficient compared to thread-parallelism.

7 Effect of using available cores subset on dense processors

7.1 Background

Two ways of looking at computing systems are through nodes which are their cost sizing blocks and through number of cores available which are their throughput sizing factors. When choosing the former and because processors have different prices, clock rates, core counts and memory bandwidth, optimizing for turnaround time or throughput may depend on running less cores than available. Since licensing costs are assessed by the number of threads or processes being run as opposed to the underlying number of cores present on the hardware, there is no licensing cost downside in not using all cores available. The deployment of threads or processes across partially used nodes should be done carefully in consideration of the existence of shared resources among cores.

7.2 Comparing Fully populated Interlagos to Half populated Interlagos



Nodes	R5.1.1 IL Full vs Half	
R5.1▸	Fully Populated No▸	Half Populated Nod
nNodes	SGI Altix ICE8400 IP106 & IP110 AMD 6282SE 2.6 Ghz - 16-core 1333Mhz	HALF POPULATED SGI Altix ICE8400 IP106 & IP110 AMD 6282SE 2.6 Ghz - 16-core 1333Mhz
1	1.40	1.13
2	2.71	2.21
4	5.05	4.03
8	8.70	7.02
16	13.53	10.18
32	19.85	16.56
64	18.60	25.72

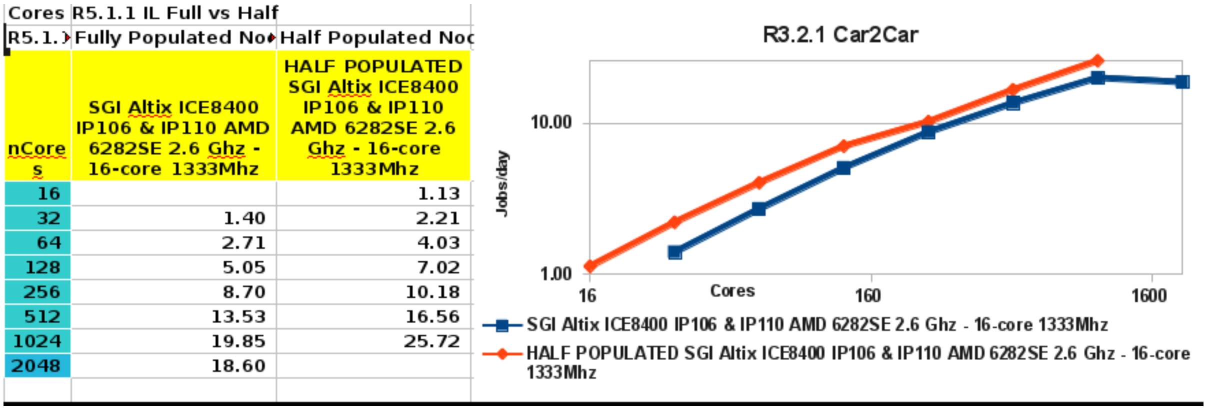
Figure 13: Node-wise comparison of fully vs. half populated Interlagos

7.2.1 Node-wise comparison

For low node numbers, fully populated Interlagos is faster than half-populated Interlagos because 16 cores per node does not use the memory bandwidth as optimally as 32 cores does (Figure 13). However, for larger number of nodes, half-populated Interlagos overtakes fully populated Interlagos because performance becomes limited by communication bandwidth. The overtaking happens for larger number of nodes on larger datasets: 8 (neon), 16 (3cars), 64 (car2car).

7.2.2 Core-wise comparison

A given number of MPI processes runs faster on half-populated nodes (Figure 14). For half-populated runs, difference between IP106-board two single port Quad Data Rate Host Card Adapters and IP110-board one dual port Quad Data Rate Host Card Adapters is barely detectable since networking is not as stressed compared to fully populated, where, as expected, two single port HCA’s (IP106) is faster than one dual port HCA (IP110) albeit by a small 6% ratio at 32 nodes.



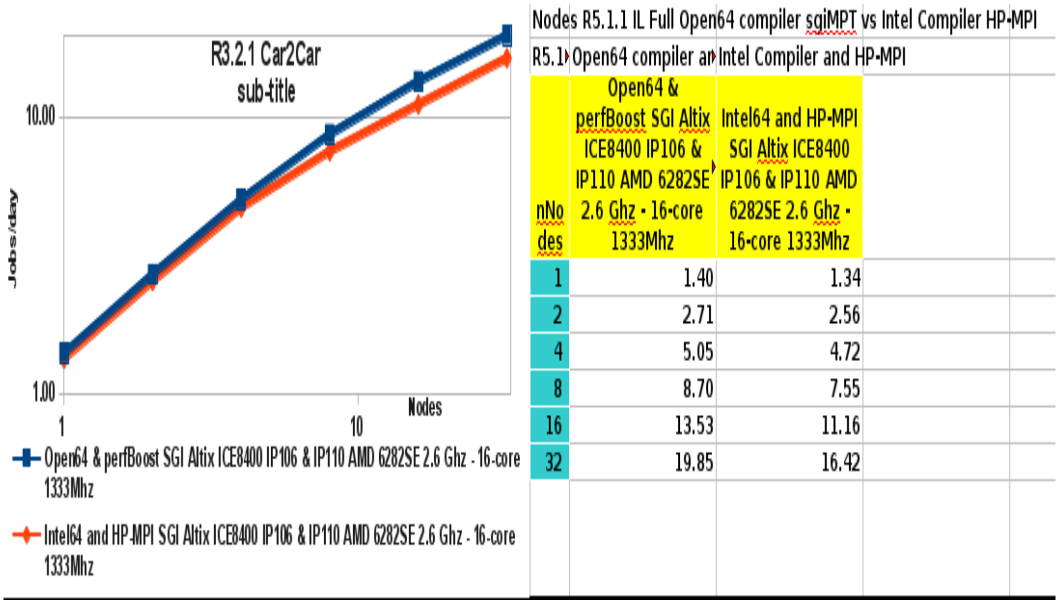
Cores R5.1.1 IL Full vs Half

R5.1.›	Fully Populated No◆	Half Populated No(
nCores	SGI Altix ICE8400 IP106 & IP110 AMD 6282SE 2.6 Ghz - 16-core 1333Mhz	HALF POPULATED SGI Altix ICE8400 IP106 & IP110 AMD 6282SE 2.6 Ghz - 16-core 1333Mhz
16		1.13
32	1.40	2.21
64	2.71	4.03
128	5.05	7.02
256	8.70	10.18
512	13.53	16.56
1024	19.85	25.72
2048	18.60	

R3.2.1 Car2Car
Jobs/day
10.00
1.00
16
Cores
160
1600
SGI Altix ICE8400 IP106 & IP110 AMD 6282SE 2.6 Ghz - 16-core 1333Mhz
HALF POPULATED SGI Altix ICE8400 IP106 & IP110 AMD 6282SE 2.6 Ghz - 16-core 1333Mhz

Figure 14: Core-wise comparison of fully vs. half populated Interlagos

7.7 Perfboost benefits



R3.2.1 Car2Car
sub-title
Jobs/day
10.00
1.00
1
10
Nodes
Open64 & perfBoost SGI Altix ICE8400 IP106 & IP110 AMD 6282SE 2.6 Ghz - 16-core 1333Mhz
Intel64 and HP-MPI SGI Altix ICE8400 IP106 & IP110 AMD 6282SE 2.6 Ghz - 16-core 1333Mhz

Nodes R5.1.1 IL Full Open64 compiler sgiMPT vs Intel Compiler HP-MPI

R5.1›	Open64 compiler an›	Intel Compiler and HP-MPI
nNodes	Open64 & perfBoost SGI Altix ICE8400 IP106 & IP110 AMD 6282SE 2.6 Ghz - 16-core 1333Mhz	Intel64 and HP-MPI SGI Altix ICE8400 IP106 & IP110 AMD 6282SE 2.6 Ghz - 16-core 1333Mhz
1	1.40	1.34
2	2.71	2.56
4	5.05	4.72
8	8.70	7.55
16	13.53	11.16
32	19.85	16.42

Figure 15: Node-wise comparison of Open64 compiler with sgi MPT vs Intel Compiler and HP-MPI

On AMD processor, Open64 compiler combined with SGI MPT through PerfBoost gives better performance than with Intel Compiler and HP-MPI (Figure 15). The breakdown of the improvements are:

```
==================Intel-compiled HP-MPI
 Elapsed:   314 sec. ( 0 hours 5 min. 14 sec.) 29977 cycles
==================Intel-compiled MPT 2.06 beta
 Elapsed:   300 sec. ( 0 hours 5 min. 0 sec.) 29977 cycles
==================AVX Beta HP-MPI
Elapsed:   212 sec. ( 0 hours 3 min. 32 sec.) 29977 cycles
==================AVX Beta PerBoost MPT 2.06 beta
Elapsed:   202 sec. ( 0 hours 3 min. 22 sec.) 29977 cycles
```

Conclusions

This study showed how providing a higher grade of a single system attribute like CPU frequency, interconnect and number of threads per process brings diminishing returns if the other attributes are kept unchanged. Therefore trades-off's exist when particular metrics such as turnaround times, throughput, costs in acquisition, license, energy, facilities, maintenance to minimize are chosen.

Attributions

LS-DYNA is a registered trademark of Livermore Software Technology Corp. SGI, ProPack and Cyclone are registered trademarks or trademarks of Silicon Graphics International Corp. or its subsidiaries in the United States or other countries. Xeon is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries. AMD and Opteron are trademarks or registered trademarks of Advanced Micro Devices, Inc. Linux is a registered trademark of Linus Torvalds in several countries. SUSE is a trademark of SUSE LINUX Products GmbH, a Novell business. All other trademarks mentioned herein are the property of their respective owners.

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. Linux c Application Tuning Guide. Silicon Graphics International, Fremont, California, 2009.
[4] John Baron and Paul Kinyon. "Selecting the Most Effective InfiniBand Topology". http://www.sgi.com/pdfs/4312.pdf , March 2011.
[5] Yih-Yih Lin and Jason Wang. "Performance of the Hybrid LS-DYNA on Crash Simulation with the Multicore Architecture". In 7th European LS-DYNA Conference, 2009.
[6] Olivier Schreiber, Scott Shaw, Brian Thatch, and Bill Tang. "LS-DYNA Implicit Hybrid Technology on Advanced SGI Architectures". http://www.sgi.com/pdfs/4231.pdf, July 2010.