Leveraging LS-DYNA Explicit, Implicit, Hybrid Technologies with SGI hardware and d3VIEW Web Portal software

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

Abstract
LSTC Explicit, Implicit solver technologies are closely integrated following LSTC’s single executable strategy. Seamless switching from large time steps transient dynamics to linear statics and normal modes analysis can thus consistently exploit latest algorithm improvements in Shared Memory Parallelism (SMP), Distributed Memory Parallelism (DMP) and their combination (Hybrid Mode) and leverage SGI computer architectures using SGI’s software stack, establishing ‘topcrunch’ world records since 2007.

This paper will show how this is accomplished on SGI’s multi-node Distributed Memory Processor clusters such as SGI® Rackable® systems and SGI® ICE™ X up to Shared Memory Processor servers such as SGI® UV™ 2000 servers. This paper will discuss how customers are using SGI’s compute and storage infrastructure to run LS-DYNA simulations using the d3VIEW™ application in a massively scalable environment.

SGI’s front-end to Cyclone is powered by d3VIEW™, a web portal based software used to submit, monitor and view results without the need to download large files. d3VIEW’s Simlyzer™ technology performs post-simulation analysis and visualization that is proven to eliminate over 80% of the LS-DYNA post processing repetitive tasks with no necessary scripting.
TABLE OF CONTENTS

1.0 Hardware Systems 3
   1.1 SGI® Rackable Cluster 3
   1.2 SGI® ICE™ X 3
   1.3 SGI® UV™ 2000 4
   1.4 Access to benchmark systems 5
   1.5 d3View 5

2.0 LS-DYNA 6
   2.1 Versions Used 6
   2.2 Parallel Processing Capabilities of LS-DYNA 6
      2.2.1 Hardware and Software nomenclature 6
      2.2.2 Parallelism background 6
      2.2.3 Parallelism metrics 7

3.0 Tuning 7
   3.1 Using only a subset of available cores on dense processors 7
   3.2 Hyperthreading 8
   3.3 MPI tasks and OpenMP thread allocation across nodes and cores 8
   3.4 SGI Performance suite MPI, PerfBoost 8
   3.5 SGI Accelerate LibFFIO 8

4.0 Benchmarks description 8
   4.1 Neon Refined Revised 8
   4.2 3 Vehicle Collision 8
   4.3 car2car 9

5.0 Absolute Performance Results 10
   5.1 Absolute performance comparison for Neon Refined Revised 11
   5.2 Absolute performance comparison for 3 Vehicle Collision 12
   5.3 Absolute performance comparison for car2car 13

6.0 Interconnect effect 14
   6.1 Interconnect effect Neon Refined Revised 14
   6.2 Interconnect effect 3 Vehicle Collision 15
   6.3 Interconnect effect car2car 15

7 Turbo, CPU Frequency effect 15

8 MPI library effect 16

9 Summary 16

10 References 16

11 Attributions 16
1.0 Hardware Systems

Various systems comprised in SGI product line and available through SGI Cyclone were used to run the benchmarks.

1.1 SGI® Rackable Cluster

SGI Rackable cluster supports up to 256GB of memory per node in a dense architecture with up to 32 cores per 1U with support for Linux®, FDR and QDR Infiniband® interconnect, eight-core processors, GPU’s and DDR3 memory (Fig.1). Configuration used for the benchmarks was:

- Intel® Xeon® 8-core 2.6 GHz E5-2670 or 6-core 2.9 GHz E5-2667
- IB QDR or FDR interconnect
- 4 GB of Memory/core
- Altair® PBSPro Batch Scheduler v11
- SLES or RHEL with latest SGI Performance Suite, Accelerate

1.2 SGI® ICE™ X

SGI ICE X integrated blade cluster is a highly scalable, diskless, cable-free infiniband interconnect high density rack mounted multi-node system. ICE X combines Intel® Xeon® processor E5-2600 series platform with a unique board and interconnect design. Running on standard Linux®, SGI ICE X delivers over 53 teraflops per rack of 2,304 processor cores (Fig. 2). Configuration used for the benchmarks was:

- Intel® Xeon® 8-core 2.6 GHz E5-2670 or 6-core 2.9 GHz E5-2667
- Integrated IB FDR interconnect Hypercube/Fat Tree
- 4 GB of Memory/core
- Altair® PBSPro Batch Scheduler v11
- SLES or RHEL with latest SGI Performance Suite, Accelerate
1.3 SGI® UV™ 2000

SGI UV 2000 scales up to 256 sockets (2,048 cores, 4096 threads) with architectural support for up to 262,144 cores (32,768 sockets). Support for up to 64TB of global shared memory in a single system image enables SGI UV to be very efficient for applications ranging from in-memory databases, to a diverse set of data and compute-intensive HPC applications. It is simpler with this platform for the user to access large resources with programming via a familiar OS [1], without the need for rewriting software to include complex communication algorithms. TCO is lower due to its low, one-system administration demands.

CAE workflow can be accelerated for overall time to solution by running pre/Post-processing, solvers and visualization on one machine without moving data (Fig. 3). Flexibility of sizing memory allocated to a job independently from the core allocation in a multi-user, heterogenous workload environment prevents jobs requiring a large amount of memory from being starved for cores. For example, a job requiring 128GB to run in-core could be broken up through domain decomposition into 8 parallel MPI processes needing only 16GB so one could run it on 8 24GB cluster nodes. But these 8 cluster nodes may not be available in a busy environment so the job would be waiting in the queue, effectively starved for nodes. On the Shared Memory Parallel system, one can always find 8 free cores and allocate the 128GB to them for the job and there is also the option to run the job serially on 1 core with 128GB allocation.
Configuration used for the benchmarks was:

- 64 sockets (512 cores) per rack
- Intel® Xeon® 8 core 2.4 GHz E5-4640 or 6 core 2.9 GHz E5-4617
- SGI NUMAlink® 6 Interconnect
- 4 GB of Memory/core
- Altair® PBSPro Batch Scheduler with CPUESET MOM v11
- SLES or RHEL with latest SGI Performance Suite, Accelerate

1.4 Access to benchmark systems

SGI offers Cyclone computing resources to all SGI advanced architectures aforementioned (Fig. 4). Cyclone™ services can reduce customer time to results 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.

1.5 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 portal software is the industry leader in providing a platform for simulation engineers in the area of simulation data visualization and collaboration.

d3VIEW portal software can be integrated with SGI clusters to provide users an instant access for running complex simulations. Jobs can be submitted and monitored from any internet-enabled device. d3VIEW portal software 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 SGI clusters.
Once the job completes, d3VIEW portal software 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.

2.0 LS-DYNA

2.1 Versions Used

LS-DYNA/MPP ls971 R5.1.1 hybrid with Message Passing Interface or R3.2.1. The latter is faster than R5.1.1 by 25% (neon) to 35% (car2car) because at R4.2.1, coordinate arrays where coded to double precision for the simulation of finer time-wise phenomena.

2.2 Parallel Processing Capabilities of LS-DYNA

2.2.1 Hardware and Software Nomenclature

Specific terminology is adopted differentiating processors and cores in hardware:

- **Core**: a Central Processing Unit (CPU) capable of arithmetic operations.
- **Processor**: a four (quad-core), six (hexa-core), eight or twelve core assembly socket-mounted device.
- **Node or Host**: a computer system associated with one network interface and address. With current technology, it is implemented on a board in a rack-mounted chassis or blade enclosure. The board comprises two sockets or more.

On the software side one distinguishes between:

- **Process**: execution stream having its own address space.
- **Thread**: execution stream sharing address space with other threads.

Based on these definitions, it follows there is not necessarily one to one mapping between processes and cores when describing a computational run.

2.2.2 Parallelism Background

Parallelism in scientific/technical computing exists in two paradigms implemented separately or recently combined in the so-called Hybrid codes: Shared Memory Parallelism (SMP) appeared in the 1980’s with the strip mining of ‘DO loops’ and subroutine spawning via memory-sharing threads. In this paradigm, parallel efficiency is affected by the ratio of arithmetic operations versus data access referred to as ‘DO loop granularity’. In the late 1990’s Domain Decomposition Parallel (DMP) Processing was introduced and proved more suitable for performance gains because of its coarser grain parallelism based on geometry, matrix or frequency domain decomposition. It consolidated on the MPI Application Programming Interface. In this paradigm, parallel efficiency is affected by the boundaries created by the partitioning. In the mean time, Shared Memory Parallelism saw adjunction of mathematical libraries already parallelized using efficient implementation through Shared Memory Parallelism API OpenMPTM (Open Multi-Processing) and Pthreads standards. These two paradigms run 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.

The two methods can be combined together in what is called ‘Hybrid Mode’.
It has to be noted that while Shared Memory Processing cannot span cluster nodes both communication and memory-wise, Distributed Memory Parallelism can also be used within a Shared Memory system. Since DMP has coarser granularity than SMP, it is preferable, when possible to run DMP within Shared Memory systems [2],[3].

### 2.2.3 Parallelism Metrics

Amdahl’s Law, ‘Speedup yielded by increasing the number of parallel processes of a program is bounded by the inverse of its sequential fraction’ is also expressed by the following formula where P is the program portion that can be made parallel, 1-P is its serial complement and N is the number of processes applied to the computation:

\[
\text{Amdahl Speedup} = \frac{1}{(1-P) + P/N}
\]

A derived metric thus is:

\[
\text{Efficiency} = \frac{\text{Amdahl Speedup}}{N}
\]

A trend can already be deduced by the empirical fact that the parallelizable fraction of an application is more dependent on CPU speed, and the serial part, comprising overhead tasks is more dependent on RAM speed or I/O bandwidth. Therefore, a higher CPU speed system will have a larger 1-P serial part and a smaller P parallel part causing the Amdahl Speedup to decrease. This can lead to misleading assessment of different hardware configurations as shown by this example:

<table>
<thead>
<tr>
<th>N</th>
<th>System A elapsed seconds</th>
<th>System B elapsed seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1000</td>
<td>640</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>80</td>
</tr>
<tr>
<td>Speedup</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>

where System A and System B parallel speedups are 10 and 8, respectively, even though System B has faster raw performance. Normalizing speedups with the slowest system serial time remedies this problem:

| Speedup | 10 | 12.5 |

Two other useful notions used for ranking supercomputers especially are:

- Strong scalability: Decreasing execution time on a particular dataset by increasing processes count.
- Weak scalability. Keeping execution time constant on ever larger datasets by increasing processes count.

It may be preferable, in the end, to instead use a throughput metric, especially if several jobs are running simultaneously on a system:

\[
\text{Number of jobs/hour/system} = \frac{3600}{\text{Job elapsed time}}
\]

The system could be a chassis, rack, blade, or any number of units of hardware provisioned indivisibly.

### 3.0 Tuning

#### 3.1 Using only a subset of available cores on dense processors

Two ways of looking at computing systems are either through nodes which are their cost sizing blocks or through cores available which are their throughput sizing factors. When choosing the former, because processors have different prices, clock rates, core counts and memory bandwidth, optimizing for turnaround time or throughput will depend on running on all or a subset of cores available. Since licensing charges are assessed by the number of threads or processes being run as opposed to the actual number of physical cores
present on the system, 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. For this study, however, this second strategy is not shown here.

### 3.2 Hyperthreading

Beyond 2 nodes, with LS-DYNA, hyperthreading gains are negated by added communication costs between the doubled numbers of MPI processes. These results are not shown here.

### 3.3 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, memory2**: Size in words of allocated RAM for MPI processes. (A word is 4 or 8 bytes long for single or double precision executables, respectively.)

### 3.4 SGI Performance suite MPI, PerfBoost

The ability to bind an MPI rank to a processor core is key to control performance because of the multiple node/socket/core environments. From [4], ‘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, SGI MPI, 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™ software. PerfBoost provides a Platform-MPI, IntelMPI, OpenMPI, HP-MPI ABI-compatible interface to SGI MPI. However, since SGI MPI native executables are available from LSTC, PerfBoost is not necessary.

### 3.5 SGI Accelerate LibFFIO

LS-DYNA/MPP/Explicit is not I/O intensive and placement can be handled by SGI MPI, therefore, libFFIO is not necessary.

### 4.0 Benchmarks Description

The benchmarks used are the three TopCrunch (http:www.topcrunch.org) dataset--created by National Crash Analysis Center (NCAC) at George Washington University. The TopCrunch project was initiated to track aggregate performance trends of high performance computer systems and engineering software. Instead of using a synthetic benchmark, an actual engineering software applications, LS-DYNA/Explicit is used with real data. Since 2007, SGI has held the top performing position on the three datasets. The metric is: Minimum Elapsed Time and the rule is that all cores for each processor must be utilized.

### 4.1 Neon Refined Revised

The benchmark consists of a vehicle based on 1996 Plymouth Neon crashing with an initial speed 31.5 miles/hour. The model comprises 535k elements, 532,077 shell elements, 73 beam elements, 2,920 solid elements, 2 contact interfaces, 324 materials. The simulation time is 30 ms (29,977 cycles) (figure 5) and writes 68,493,312 Bytes d3plot and 50,933,760 Bytes d3plot[01-08] files at 8 time steps from start to end point (114MB).
4.2 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 (figure 6) 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), writing 65,853,440 Bytes d3plot and 33,341,440 Bytes d3plot[01-19] files at 20 time steps from start to end point (667MB). 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 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.

4.3 car2car

The benchmark consists of an angled 2 vehicle collision (figure 7). The vehicle models are based on NCAC minivan model with 2.5 million elements. The simulation writes 201,854,976 Bytes d3plot and 101,996,544 Bytes d3plot[01-25] files at 26 time steps from start to end point (2624MB).
## 5.0 Absolute Performance Results

Figure 8 shows a table with the relevant characteristics listed to properly compare the performance data obtained on the benchmark systems or on published topcrunch.org data. Within each system, it is possible to scale CPU frequency to further evaluate performance (Section 7). A case by case look at the results follows in the next subsections. The number of MPI processes chosen for each dataset are 256, 512 and 1024, corresponding to peak parallel efficiency.

<table>
<thead>
<tr>
<th>Server Name</th>
<th>cy007</th>
<th>cy007</th>
<th>cy002</th>
<th>cy002</th>
<th>cy002</th>
<th>cy022</th>
<th>cy022</th>
<th>cy022</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue</td>
<td>T2600</td>
<td>T2600</td>
<td>T2900</td>
<td>T2700</td>
<td>T2600</td>
<td>T2701</td>
<td>T2700</td>
<td>T2600</td>
</tr>
<tr>
<td>Vendor</td>
<td>SGI</td>
<td>SGI</td>
<td>SGI</td>
<td>SGI</td>
<td>SGI</td>
<td>SGI</td>
<td>SGI</td>
<td>SGI</td>
</tr>
<tr>
<td>Platform</td>
<td>Rackable</td>
<td>Rackable</td>
<td>ICE X</td>
<td>ICE X</td>
<td>ICE X</td>
<td>UV 2000</td>
<td>UV 2000</td>
<td>UV 2000</td>
</tr>
<tr>
<td>Processor Vendor</td>
<td>Intel®</td>
<td>Intel®</td>
<td>Intel®</td>
<td>Intel®</td>
<td>Intel®</td>
<td>Intel®</td>
<td>Intel®</td>
<td>Intel®</td>
</tr>
<tr>
<td>Processor Brand</td>
<td>Xeon®</td>
<td>Xeon®</td>
<td>Xeon®</td>
<td>Xeon®</td>
<td>Xeon®</td>
<td>Xeon®</td>
<td>Xeon®</td>
<td>Xeon®</td>
</tr>
<tr>
<td>Processor Model</td>
<td>E5-2670</td>
<td>E5-2670</td>
<td>E5-2670</td>
<td>E5-2670</td>
<td>E5-2670</td>
<td>E5-2670</td>
<td>E5-2670</td>
<td>E5-2670</td>
</tr>
<tr>
<td>Clock Speed (GHz)</td>
<td>2.60</td>
<td>2.60</td>
<td>2.90</td>
<td>2.70</td>
<td>2.60</td>
<td>2.70</td>
<td>2.70</td>
<td>2.60</td>
</tr>
<tr>
<td>turbo</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>RAM Speed (MHz)</td>
<td>1600</td>
<td>1600</td>
<td>1600</td>
<td>1600</td>
<td>1600</td>
<td>1600</td>
<td>1600</td>
<td>1600</td>
</tr>
<tr>
<td>Cores/Socket</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Sockets/Node</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>Cores/Node</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>512</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>Memory/Node (GB)</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>4096</td>
<td>4096</td>
<td>4096</td>
</tr>
<tr>
<td>Memory/Core (GB)</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Storage (Not used)</td>
<td>SAIA</td>
<td>1TB</td>
<td>7.2 RPM</td>
<td>Diskless</td>
<td>Diskless</td>
<td>Diskless</td>
<td>IS5500 RAID5</td>
<td></td>
</tr>
<tr>
<td>Interconnect</td>
<td>IB QDR - 4x</td>
<td>IB QDR - 4x</td>
<td>IB FDR - 4x</td>
<td>IB FDR - 4x</td>
<td>IB FDR - 4x</td>
<td>NUMAlink</td>
<td>NUMAlink</td>
<td>NUMAlink</td>
</tr>
<tr>
<td>Bandwidth (Gb/s)</td>
<td>40</td>
<td>40</td>
<td>56</td>
<td>56</td>
<td>56</td>
<td>53</td>
<td>53</td>
<td>53</td>
</tr>
<tr>
<td>Latency (usec)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Nodes</td>
<td>64</td>
<td>64</td>
<td>144</td>
<td>144</td>
<td>144</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>Sockets</td>
<td>128</td>
<td>128</td>
<td>288</td>
<td>288</td>
<td>288</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>Cores</td>
<td>1024</td>
<td>1024</td>
<td>2304</td>
<td>2304</td>
<td>2304</td>
<td>512</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>OS</td>
<td>SLES11SP</td>
<td>SLES11SP</td>
<td>SLES11SP</td>
<td>SLES11SP</td>
<td>SLES11SP</td>
<td>SLES11SP</td>
<td>SLES11SP</td>
<td>SLES11SP</td>
</tr>
<tr>
<td>MPI Library</td>
<td>MPT 2.07e</td>
<td>MPT 2.07e</td>
<td>MPT 2.07e</td>
<td>MPT 2.07e</td>
<td>MPT 2.07e</td>
<td>MPT 2.07e</td>
<td>MPT 2.07e</td>
<td>MPT 2.07e</td>
</tr>
<tr>
<td>Label</td>
<td>Rackable</td>
<td>Rackable</td>
<td>ICE X 2.9</td>
<td>ICE X 2.7</td>
<td>ICE X 2.6</td>
<td>UV 2000</td>
<td>UV 2000</td>
<td>UV 2000</td>
</tr>
<tr>
<td>cars @ 256 cores</td>
<td>60</td>
<td>68</td>
<td>57</td>
<td>62</td>
<td>64</td>
<td>70</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>cars @ 512 cores</td>
<td>431</td>
<td>488</td>
<td>427</td>
<td>449</td>
<td>460</td>
<td>529</td>
<td>555</td>
<td>569</td>
</tr>
<tr>
<td>cars @ 1024 cores</td>
<td>1887</td>
<td>2122</td>
<td>1869</td>
<td>1998</td>
<td>2054</td>
<td>2485</td>
<td>2485</td>
<td>2485</td>
</tr>
</tbody>
</table>

*Figure 8: Global table of computed or previously published data for various systems*
5.1 Absolute performance comparison for Neon Refined Revised

Figure 9 shows that new Intel Xeon E5-2600 processor running at 2.6 GHz with Turbo Boost enabled outperforms previous generation Intel Xeon EP X5690 processor even though frequency is lower. At same 2.6GHz frequency, ICE X cluster increase performance over Rackable cluster by 6% because of its FDR Infiniband interconnect and predictably, at 2.9 GHz ICE X dominates all platforms. SGI UV 2000 shared memory system performance is in line with the Rackable cluster as it uses almost the same processor as opposed to previous generation UV 1000's Intel Xeon EX E7-8837 processors.

![Elapsed time comparisons between platforms, neon refined revised](image)
5.2 Absolute performance comparison for 3 Vehicle Collision

Figure 10 shows that new Intel Xeon E5-2600 processor running at 2.6 GHz with Turbo Boost enabled outperforms previous generation Intel Xeon EP X5690 processor even though frequency is lower. The ICE X cluster dominates all platforms at any frequency because of its FDR Infiniband interconnect. UV 2000 performance is in line with Rackable as it uses almost the same processor as opposed to previous generation UV 1000’s Intel Xeon Westmere EX E7-8837.

![Figure 10: Elapsed time comparisons between platforms, 3 vehicle collision](image-url)
5.3 Absolute performance comparison for car2car

Figure 11 shows that new Intel Xeon E5-2600 processor running at 2.6 GHz with Turbo Boost enabled outperforms previous generation Intel XeonEP X5690 processor even though frequency is lower. ICE X dominates all platforms at any frequency because of its FDR Infiniband interconnect.
6.0 Interconnect effect

SGI Performance Suite MPInside, a MPI profiling and performance analysis tool that provides finer-grained metrics for analyzing MPI communications [5] was used to separate timings imputed to computational work and communications. A typical chart is shown in Figure 12 where Computation work is the bottom blue layer.

![Figure 12: Typical MPInside chart.](image)

6.1 Interconnect effect Neon Refined Revised

From left to right, Figure 13 shows that for same CPU frequency of 2.60 GHz, communication-wise, the Rackable servers with QDR is slower than the ICE X system with FDR by 6% and the UV 2 server with NUMAlink® interconnect 6 shows higher communication times (12%) while the UV 1 server with NUMAlink interconnect 5 also shows higher computation times for a combined 31% slow down.

![Figure 13: Rackable with QDR, ICE X with FDR, UV2 with NL6, UV1 with NL5](image)
6.2 Interconnect effect 3 Vehicle Collision

From left to right, Figure 14 shows that for same CPU frequency 2.60 GHz, communication-wise, the Rackable servers with QDR is slower than the ICE X system with FDR by 6% but faster than the UV 2 server with NUMAlink® 6 interconnect and UV 1 with NUMAlink® 5 by 17%.

![Figure 14: Rackable with QDR, ICE X with FDR, UV2 with NL6, UV1 with NL5](image)

6.3 Interconnect effect car2car

From left to right, Figure 15 shows that for same CPU frequency 2.60 GHz, communication-wise, the Rackable servers with QDR is slower than the ICE X system FDR by 3% (UV server times not available at time of study).

![Figure 15: Rackable with QDR, ICE X with FDR](image)

7 Turbo, CPU Frequency effect

From left to right, Figure 16 shows for car2car that the Rackable server with Turbo ON is 12% faster than Turbo OFF at 2.6 GHz. ICE X at 2.6 GHz is 2.7% slower than at 2.7 GHz and 9% slower than at 2.9 GHz. Figure 17 shows the percentages increase in performance for the 3 cases compared with ideal values. One can see that changes in CPU frequency do not translate in the same percentage increase of performance.

![Figure 16: Rackable 2.6GHz Turbo Boost ON, Turbo Boost OFF, ICE X 2.6, 2.7 2.9GHz](image)
8 MPI library effect

As mentioned in section 3.4, and shown by the following elapsed seconds (lower is better) table, performance can increase by using SGI MPI and tuning may affect results as well:

<table>
<thead>
<tr>
<th>Dataset \ MPI</th>
<th>SGI MPI</th>
<th>Platform MPI</th>
<th>Intel MPI</th>
<th>Source: Topcrunch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neon Refined Revised</td>
<td>60</td>
<td>71</td>
<td>81</td>
<td>64 (Intel MPI)</td>
</tr>
<tr>
<td>3 Vehicle Collision</td>
<td>431</td>
<td>514</td>
<td>595</td>
<td>530 (Platform MPI)</td>
</tr>
</tbody>
</table>

9 Summary

Upgrading a single system attribute like CPU frequency, interconnect, number of cores per node, RAM speed, brings diminishing returns if the others are kept unchanged. Trades can be made based on metrics such as dataset turnaround times or throughput, acquisition, licensing, energy, facilities, maintenance costs to minimize.

10 References


11 Attributions

LS-DYNA, is a registered trademark of Livermore Software Technology Corp. SGI, Rackable, NUMAlink, SGI Ice X, SGI UV, 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. 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.

Global Sales and Support: sgi.com/global