TECH GUIDE

# SGI® Technology Guide for Users of Abaqus®

September, 2014


**Authors**

Scott Shaw†, Dr. Olivier Schreiber†, Tony DeVarco††

†Senior CAE Applications Engineer, SGI Applications Engineering

††Director of SGI Virtual Product Development Solutions



Published in cooperation with Dassault Systèmes SIMULIA

# TABLE OF CONTENTS

# 1.0 Purpose

This SGI technology guide is intended to help customers make knowledgeable choices in regards to selecting high-performance computing (HPC) hardware to optimally run Abaqus software from the SIMULIA brand of Dassault Systèmes. This guide reviews the performance of Abaqus executed on three types of SGI platforms: the SGI® Rackable™ cluster, the SGI® ICE™ X cluster and the SGI® UV™ 2000 Shared Memory Parallel (SMP) platform. In addition to presenting performance results on these three computer platforms, we discuss the benefits of using multicore Intel® processors, the trade-offs of different network topologies, NVIDIA® compute GPU device performance and the use of SGI® MPI PerfBoost. Also included are sizing guidelines and recommendations for HPC computing platforms running Abaqus.

The Abaqus models selected for this guide are included with each Abaqus release. Using the common datasets provides a way to characterize system performance on various platforms to allow general performance comparisons.

# 2.0 Abaqus Portfolio

Abaqus from SIMULIA, the Dassault Systèmes brand for realistic simulation, is an industry-leading product family that provides a comprehensive and scalable set of Finite Element Analysis (FEA), multiphysics solvers and modeling tools for simulating a wide range of linear and nonlinear model types. It is used for stress, heat transfer crack initiation, failure and other types of analysis in mechanical, structural, aerospace, automotive, bio-medical, civil, energy, and related engineering and research applications. Abaqus includes four core products: Abaqus/CAE, Abaqus/Standard, Abaqus/Explicit, and Abaqus/CFD. This guide will focus mainly on Abaqus/Standard and Abaqus/Explicit running on SGI systems.

# 3.0 Abaqus Pre and Post Processor

Abaqus/CAE provides users with a modeling and visualization environment for Abaqus analysis. With direct access to CAD models, advanced meshing and visualization, Abaqus/CAE is the modeling environment of choice for Abaqus users. Once your model has been created and a mesh is generated, an input file is created which contains the model data and history data. Model data includes the elements, element properties, material definitions, etc. Abaqus/CAE is commonly used for preprocessing the input of files, job submittal analysis, and post processing of analysis models.

# 4.0 Parallel Processing Capabilities of Abaqus

## 4.1 Computing Nomenclature

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

## 4.2 Parallelism

Shared Memory Parallelism (SMP) appeared in the 1980s around DO loop processing or subroutine spawning and consolidated on the Open Multi-Processing Application Programming Interface (OpenMP) and POSIX Pthreads standards. Parallel efficiency is affected by the ratio of arithmetic floating point operations versus data access.

Distributed Memory Parallelism (DMP) appeared in the late 1990s 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.

- SMP is based on intra-node communication using memory shared by all cores. A cluster is made up of SMP compute nodes but each node cannot communicate with each other so scaling is limited to a single compute node.

- DMP programming model is based on MPI communications which allows all application threads to communicate with other compute nodes.
- Hybrid MPI+OpenMP bridges the gap between compute nodes allowing nodes in a cluster to communicate with each other while spawning OpenMP or Pthreads for loop processing. This combined programming model reduces the amount of message traffic on the network interconnect, and in most cases yields better performance.

Since DMP is of a coarser granularity than SMP, it is preferable, when possible, to run DMP within Shared Memory Systems. Depending on the analysis type Abaqus/Standard exploits parallelism based on MPI (DMP), OpenMP (SMP) or a combined Hybrid MPI+ OpenMP approach while Abaqus/Explicit is pure MPI.

Following is a breakdown of Abaqus programming models:

- Explicit operations are done with MPI
- Element operations are done with MPI and Threads
- Iterative solver uses MPI
- Direct Sparse solver uses MPI and Threads
- AMS and Lanczos solver uses Threads

## 5.0 Abaqus Problem Types

Abaqus runtimes are influenced by the following factors. Figure 1, represents a chart based on job turnaround time as the model sizes grow.

- Abaqus/Standard linear and nonlinear implicit solver is based on degrees of freedom (DOF) and iterations. Increasing the DOF, elements and iteration count for a simulation will influence the model's runtime.
- Abaqus/Explicit is based on the number of elements and time duration of events. Increasing the number of elements or the time duration of the simulation will influence the model's runtime.

## 6.0 Executing Abaqus with a Batch Scheduler

The batch scheduler job script is responsible for defining the necessary computing resources required for running each Abaqus analysis. Key resources to define are:

- Range of compute nodes and number of MPI tasks per node to use for the analysis
- Placement of application threads across sockets within nodes and neighboring nodes
- Control of process memory allocation to stay within node capacity
- Pre and Post staging of scratch and analysis result files

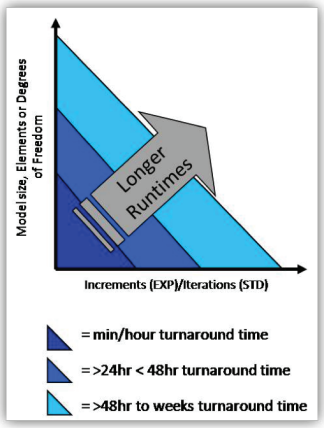



*Figure 1: Abaqus Runtime Factors*

Batch schedulers/resource managers dispatch jobs from a front-end login or through a Web GUI portal 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 shared parallel file system.

The high performance file system could be an in-memory (RAM), local drive, parallel or network attached storage file systems. In a diskless computing environment, the two choices commonly used are parallel file systems and network attached storage.

## 7.0 SGI Hardware Used For Benchmarks

High Performance Computing Platform Descriptions

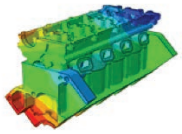
| SGI Platform | SGI® Rackable® Cluster | SGI® ICE™ X | SGI® UV™ 2000 |
|---|---|---|---|
| **Processors Type & Clock Speed** | Intel® Xeon® E5-2697 v2 2.70 GHz | Intel® Xeon® E5-2690 v2 3.00 GHz | Intel® Xeon® CPU E5-4627 3.30 GHz |
| **Total Cores/Node** | 24 | 20 | 16 (total 512 cores) |
| **Memory per Node** | 128GB | 64GB | 4TB |
| **Local Storage** | 3x SATA 1TB 7.2 RPM 3Gb/s Drive | Diskless & Lustre PFS Storage | IS5600 RAID6 Storage |
| **Network Interconnect** | IB FDR 4x | IB FDR 4x | NUMALink 6 |
| **Number of Nodes** | 32 Compute Nodes | 144 Compute Blades | 32 Compute Blades (SMP) |
| **Operating System** | SLES11 SP3 | SLES11 SP3 | SLES11 SP3 |

## 8.0 Software Environment

- SGI® Performance Suite
- Abaqus product suite 6.13 or above
- Intel® Parallel/Cluster Studio XE with Intel® Math Kernel Library (Intel® MKL)
- Altair PBS Professional® Batch Scheduler v12 or later

## 9.0 Benchmark Models

To capture performance of the various SGI platforms, we used three Abaqus models to demonstrate processor performance, memory bandwidth, disk performance and communications.



- Abaqus/Standard S4B, Cylinder head bolt-up, 5M degrees of freedom (DOF) with 5 iterations. This benchmark is a mildly nonlinear static analysis that simulates bolting a cylinder head onto an engine block. The S4B model is a compute-bound and memory bandwidth limited example which is dominated by large floating point operations per iterations 1.03E+13. The majority of the compute time is spent in the solver.

- Abaqus/Standard S6, Tire Footprint, 730K DOF with 177 iterations. This benchmark is a strongly nonlinear static analysis that determines the footprint of an automobile tire. The S6 model is communication bound vs. compute due to the low floating point operations per iteration, 4.99E+10.

- Abaqus/Explicit E6. This benchmark consists of a large number of concentric spheres with clearance between each sphere. All of the spheres are placed into a single general contact domain and the outer sphere is violently shaken which results in complex contact interactions between the contained spheres. The E6 model is an example of a memory bandwidth bound problem.

## 10.0 Results and Discussions

When measuring the total runtime of each dataset we accumulate the time in seconds for each step in the input file preprocessor, solver and postprocessor phases. Next, we take the total runtime and convert it into a rating value based on the number of analyses run during a 24 hour period, defined as “jobs per day.” Using the “jobs per day” rating system provides better multi-core and multi-node scaling guidance when comparing similar system architectures.

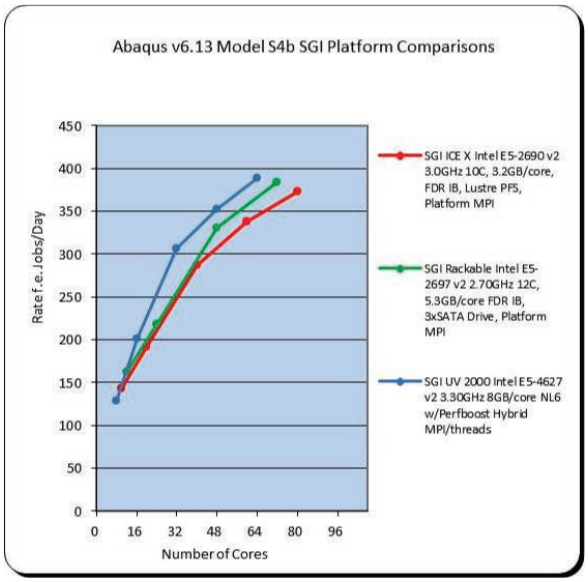
When analyzing Abaqus datasets, it is a good practice to review the output files to better understand if the analysis is computational, memory, communication or I/O bound. The majority of this information can be found in the output files generated by the analysis or by using the datacheck option. A good place to start is to review the Memory Estimates and the Problem Size in the output *.dat file. Another best practice is to check the solver wall times per iteration for each step to make sure the time delta between iterations for each step does not vary more than 60 seconds. We typically monitor the solver wall times to make sure system resources are efficiently being utilized. In a case where there is a large variance and unpredictable high values, we recommend monitoring the system resources like memory usage, Linux® page cache usage, disk activity and any enterprise system monitoring daemons. From time to time, we find customer sites with system resource monitoring, which are a bit aggressive and starve system resources from high performance computing applications resulting in slower turnaround times. As a consideration for production use, we recommend the minimum interval to check system resources while an enterprise system monitoring package is better used for minute intervals.

## 11.0 Abaqus/Standard S4B

As previously mentioned the S4B model with 5M DOF is a computationally bound analysis with a high degree of floating point operations per iteration. If the solver is dominated by computational processing with floating point operations greater than 1E+12, we find these types of models will scale better than a communication bound analysis, like the Abaqus S6 dataset, where more time is generated in communication of messages versus time spent in the solver.

In Figure 2, we evaluated the SGI ICE X, SGI Rackable and the SGI UV 2000 (SMP) server's performance running the S4B model. The chart shows the system comparisons with a rate of "jobs per day" (jobs/day) on the Y-axis and the core count used on the X-axis. Historically large SMP systems have been slower than clusters due to lower computational processor power and memory bandwidth performance. But with the UV 2000 server with Xeon® E5-4000 v2 series processors, we found above 32 cores that the UV 2000 exceeded the performance of our other two server platforms. As you review the chart below, you will observe the SGI UV 2000 server had a less job/day rating at lower core counts but the job/day rating continues to improve as the simulation scales due to the higher processor frequency with Turbo Boost enabled and the higher memory bandwidth with eight core processor per socket used. In fact, the overall UV 2000 performance increase, when compared to the other two SGI systems with higher core counts per processor, was a factor of 6-10%.

*Figure 2: System Comparisons and Job/Day Performance Rating*



When reviewing the “Number of Cores” chart comparing SGI ICE X vs. SGI Rackable in Figure 2, you will notice rating is fairly close between the two platforms, what we have observed is 8-10 cores per socket is good for Abaqus/Standard computation. The InfiniBand latency remains the same between QDR and FDR IB fabrics, but the InfiniBand bandwidth is about 30% faster over QDR 4x fabric.

The newest release of Abaqus 6.13 includes the new Intel® MKL libs that supports the Intel Xeon® E5 v2 series AVX2 extensions[1]. The new AVX2 extension 256-bit pipeline provides a 2x improvement with double precision floating point operations over the previous generations of the Intel Xeon® E5 v1 series processor architectures.

## 12.0 GPU Computing Acceleration with the S4B Dataset

GPU computing is the latest trend in high performance computing by offloading computational tasks to a GPU device which supports 400-500 floating point processing cores. Abaqus 6.11 provides support for GPU devices in Abaqus/Standard that can dramatically improve compute bound analysis by dividing computational tasks into smaller tasks that can run concurrently on the GPU device. Compute tasks normally handled by the main CPUs are now able to work more efficiently allowing access to more memory bandwidth within a compute node. In some cases, one compute node with a GPU device can be as fast as two to three compute nodes with 12 cores in each. The recent release (6.12) of Abaqus/Standard now supports multiple GPUs in compute nodes and scales over multiple GPU compute nodes within a cluster.

## 13.0 Abaqus GPU Thread Placement

One consideration to be aware of is NVIDIA® GPU PCIe bandwidth can be compromised if the GPU solver thread distance on the NUMA node exceeds a threshold to access the PCIe bus where the NVIDIA GPU card is installed. To measure the PCIe performance to and from the GPU device, we ran the SHOC[4] benchmark to capture performance.

The SHOC PCIe benchmark study is conducted on SGI Rackable compute nodes to determine if the NUMA locality was a concern as we previously discussed. We ran the SHOC tests and presented the results in Figure 3. Based on the SHOC tests on Intel® architecture we did not observe a negative impact of NUMA locality performance while testing Nvidia Tesla K40m or the Tesla K20m and the PCIe bus.

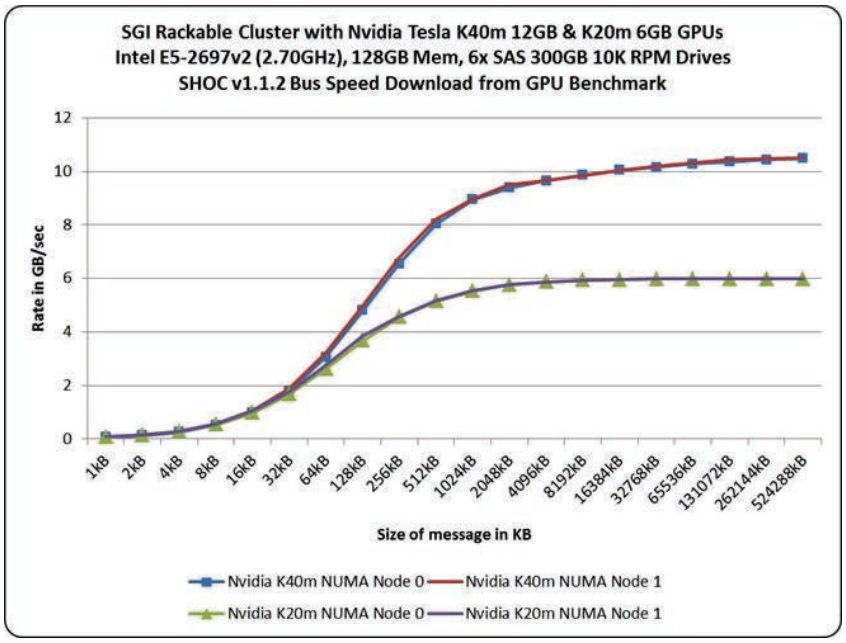



*Figure 3: Nvidia GPU NUMA Locality and PCIe Bus Performance*

## 13.1 GPU Performance with S4B

The rule of thumb we have found is if the direct sparse solver is dominated by computational processing, the GPU direct sparse solver performance will improve with floating point operations greater than one teraflop (1E+12). Obviously, there may be other conditions that improve the direct sparse solver performance with GPUs based on element types but we have found a quick reference would be the floating point operations per iteration as the key indicator. Abaqus datasets s2a (1.86E+12 FLOPs/Iteration, 475K DOF) and S4B (1.02E+13 FLOPs/Iteration, 5M DOF) are good examples of computationally intensive problems which benefit from GPU processing. In Figure 4, we present a graph showing the Jobs/day rating per node with one GPU per node.

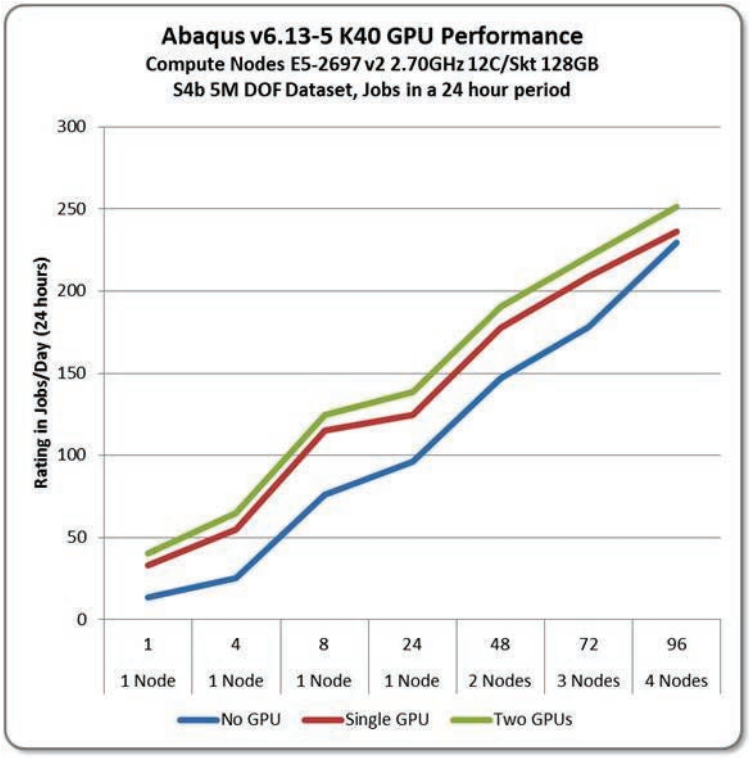



*Figure 4: Relative GPU Performance & Comparisons*

## 14.0 Abaqus/Explicit E6 Dataset and MPI Communication

With the Abaqus/Explicit E6 dataset, we take a look at the MPI communication characteristics using SGI's MPInside[5] profiling tool that examines MPI calls from an application. The MPInside profiling tool supports several MPI implementations like Intel® MPI, HP-MPI™, Platform MPI™, OpenMPI and SGI® MPI without any code modifications. MPInside can also profile POSIX file I/O system calls from each rank to better understand the file I/O patterns of Abaqus simulations. Having a better understanding of the file I/O patterns with any application will be useful when selecting the appropriate file system type such as Lustre™, NFS or local storage, or the combination of local and remote file systems.

## 14.1 MPInside Data Collection

Performing some experiments with the Abaqus/Explicit E6 model, we take a close look at the two dominating MPI calls, MPI_Iprobe and Compute as shown in Figure 5. The MPI_Iprobe MPI communication call is used for notification when MPI messages arrive, and compute is the computation portion of the code. The first stacked bar shown in Figure 5 is an experiment with "12 cores/1 node," while the second experiment is with "1 core/12 nodes."

The purpose of these two tests is to determine system resource utilization when a compute node is running with 12 cores stressing CPU, L3 cache and memory bandwidth. When running the same test with 12 nodes and one core per node, the application MPI ranks have full access to the entire L3 cache and memory bandwidth of each compute node. In the "1 core/12 nodes" test case, we explored what system resources are being utilized, and file system access patterns of each MPI rank. Based on the various tests, the Abaqus/Explicit E6 dataset is 1.4x faster in compute and only 1.08x faster in communication. Given the results of the two experiments, we can conclude the Abaqus/Explicit E6 model is sensitive to cache and memory bandwidth performance.

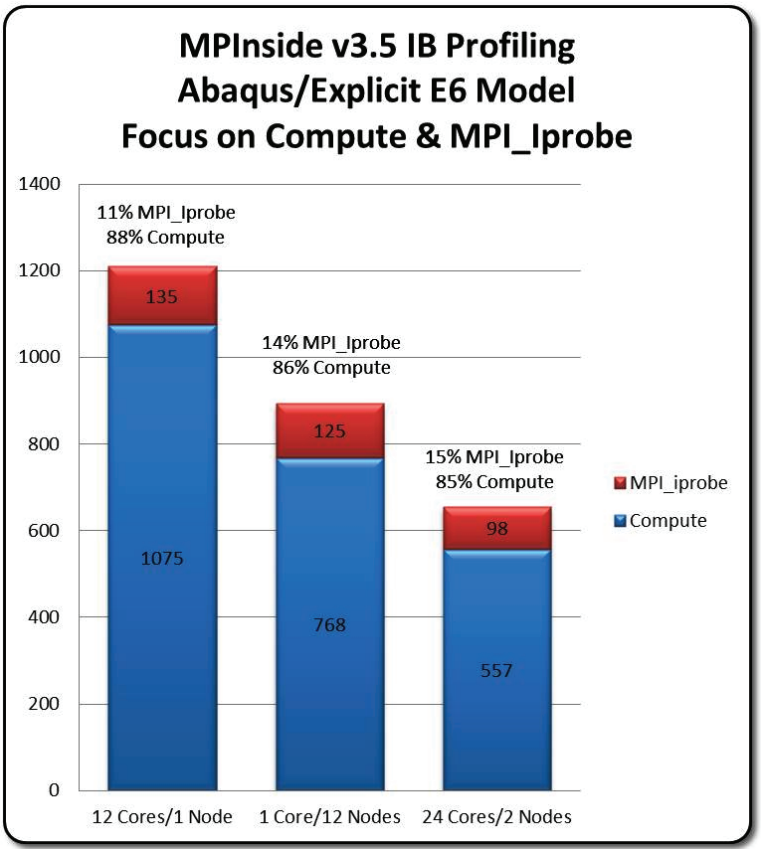



*Figure 5: MPInside Profiling Results with E6 Dataset*

## 14.2 Profiling E6 Dataset with 24 Cores

In Figure 6, we show the breakdown of MPI calls when running the Abaqus/Explicit E6 model across two 12 core compute nodes. The chart shows on the Y-axis the accumulated time, and the X-axis is the processor core ID ranging from 0 to 23. In the legend, we list all of the MPI calls that have accumulated over time. As you study Figure 6, you will notice 85% of the time is dominated by computation followed by 15% of the time dominated by monitoring MPI messages in the MPI_Iprobe call.

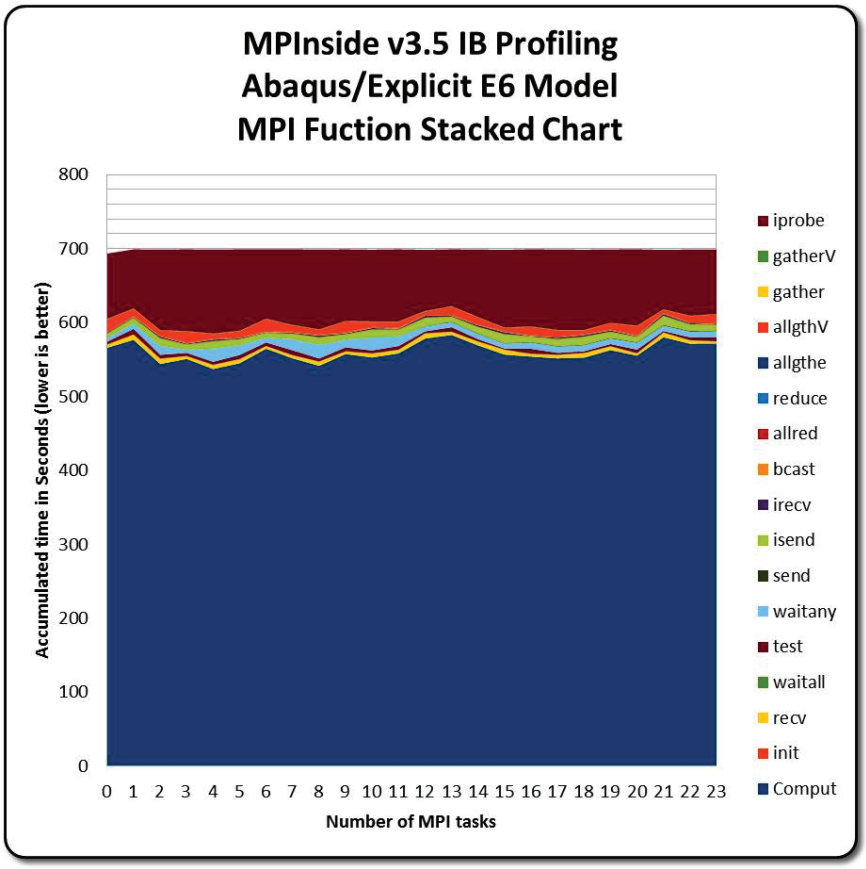



*Figure 6: MPInside Profiling Results with E6 Dataset*

## 14.3 Profiling E6 Dataset MPI Message Sizes

The MPInside profiling samples we studied included MPI message size distribution as presented in Figure 7. We observed that 47% of the MPI messages fall into a 0-128 Byte range and the total of all MPI message sizes fall in the 0-64KB range which are small message sizes.

The overall MPI communications collected was ~32GBs from all MPI ranks over 655 seconds. Based on this data collected we see about 50MB/sec just in MPI traffic alone. This amount of MPI traffic is low when InfiniBand can handle about 3.2GB/sec with QDR 4x throughput. With Gigabit Ethernet (GigE) we can easily saturate the network when running two or more E6 datasets since one analysis consumes about 57% of the total bandwidth of the GigE network. Knowing the MPI message size can be useful when evaluating new interconnect technologies.

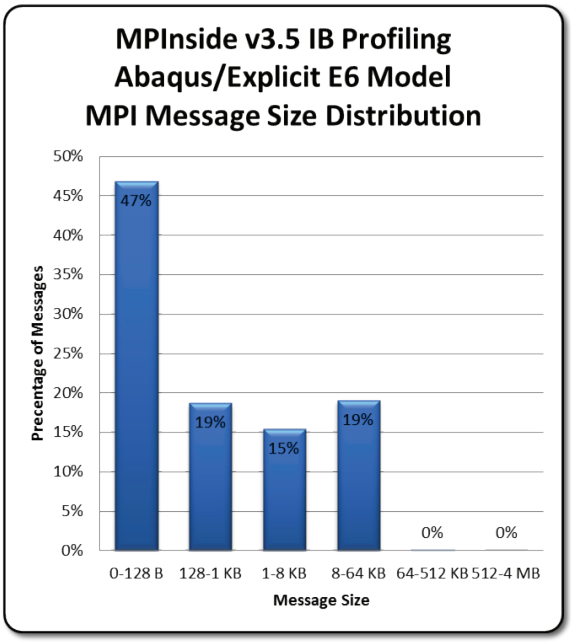



*Figure 7: MPI Message Sizes*

## 14.4 Abaqus File I/O

When studying application performance, it is good to know how I/O can influence the overall runtime and if the I/O subsystem can be the bottleneck for slow runtimes. When studying the MPInside profiled data for the E6 dataset, we observed about 84-92% of the file operations occur in the 1-8KB range, as shown in Figure 8. For most local attached storage devices, the 1-8KB transfer sizes should not be a problem but with parallel file systems that are designed for large 1MB I/O transfers like Lustre™ this would be an area of concern when selecting a parallel file system for scratch data.

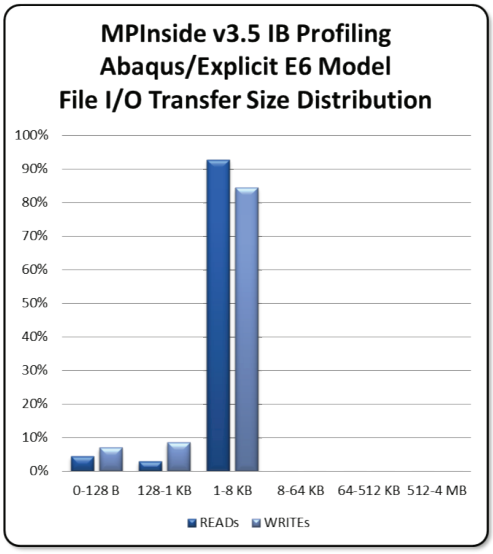



*Figure 8: MPI I/O Patterns*

## 15.0 Abaqus/Standard S6: Tire Footprint

As we previously mentioned in the dataset descriptions section, the S6 dataset was network communication bound due to heavy contact within the analysis. Without running an exhaustive set of tests to determine if the dataset is compute bound, we can review the S6 *.dat file and search for the “Memory Estimate” section to determine the floating point operations per iteration, which should be around 4.4E+10 FLOPs or 44 GigaFLOPs. Compute-bound analysis typically occur in the teraflop range and above.

## 16.0 Network Interconnect Performance with S6 Dataset

Quite often during the configuration phase of architecting a cluster based solution, the following question will arise; what network technology is best? It usually comes down to the following two types of network technologies: InfiniBand (IB) or Gigabit Ethernet (GigE). It is known that InfiniBand is faster but relatively expensive and GigE is cheaper but slower. So the choice becomes quite complex and the outcome often depends on the performance of the application on a particular network type.

Focusing on the various network interconnect performance differences, Figure 9 shows the performance differences when using a high-performance low-latency InfiniBand FDR 4x HCA vs. mid-performance high-latency GigE interconnect. When analyzing the time for the various core counts across multiple nodes, you can observe the IB time decreases as the core count increases at a higher rate than GigE recorded times. From two to four nodes InfiniBand can be 2-3x faster than GigE for MPI communications. When running these tests in a controlled environment this was the best case for the GigE performance since we used the GigE network for only MPI message passing. In a cluster where GigE was the only interconnect, the ratios will be different since the GigE network will be shared with cluster monitoring tools, user access, transfer of input/output files, NFS traffic and MPI message passing traffic. The GigE network interconnect can handle about 120 MB/sec with a 29-100 usec latency while a FDR 4x InfiniBand ConnectX3 HCA interconnect can handle about 6.8GB/sec with a 1 usec latency. Utilizing the IPoIB (TCP over InfiniBand) TCP/IP layer performance can peak about 640 MB/sec. In SGI’s Global Benchmark Center for Rackable and ICE X clusters we use IPoIB TCP/IP layer for NFS traffic across four InfiniBand network ports. Using more than one network port allows better distribution of NFS traffic and the aggregate performance can reach ~2.5GB/sec with four InfiniBand ports.

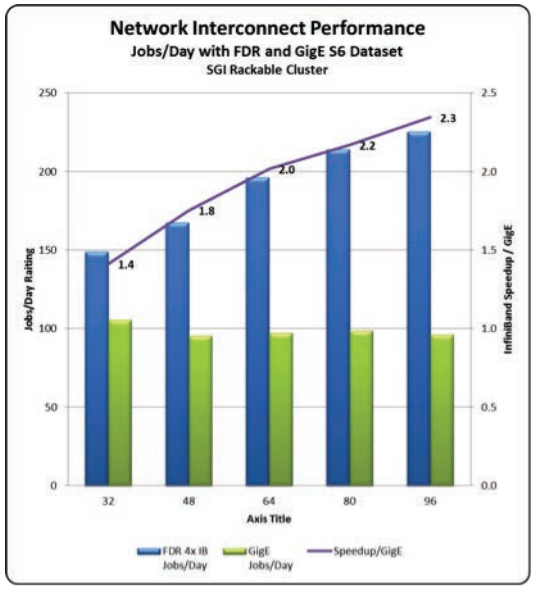



*Figure 9: Network Interconnect Performance*

The GigE performance will degrade significantly when concurrent analyses are running on a cluster which could lead to unpredictable turnaround times and an inefficient computing platform for a mixed HPC computing environment.

## 17.0 Advantages of the SGI MPI Library Through SGI PerfBoost

A key feature in any MPI implementation is the capability to bind an MPI/OpenMP application process threads to a processor core since you want to prevent the migration of application processes. When an application process migrates throughout the compute node, memory references can become scattered and more time can be spent in readback of the memory references. HP-MPI™ which has been replaced by Platform MPI™ currently provides limited CPU-affinity and core-placement support for MPI ranks, but not for SMP threads spawned by the MPI ranks.

With a combination of SGI PerfBoost and SGI MPI, the omplace command uniquely provides convenient placement of hybrid MPI/OpenMP processes within each node. The MPI library is linkless and available through the PerfBoost facility bundled with SGI Performance Suite. The Perfboost shim library only intercepts and optimizes MPI calls while other application routines execute without intervention. The supported MPI implementations are Platform MPI (HP-MPI), IntelMPI, OpenMPI, and MPICH.

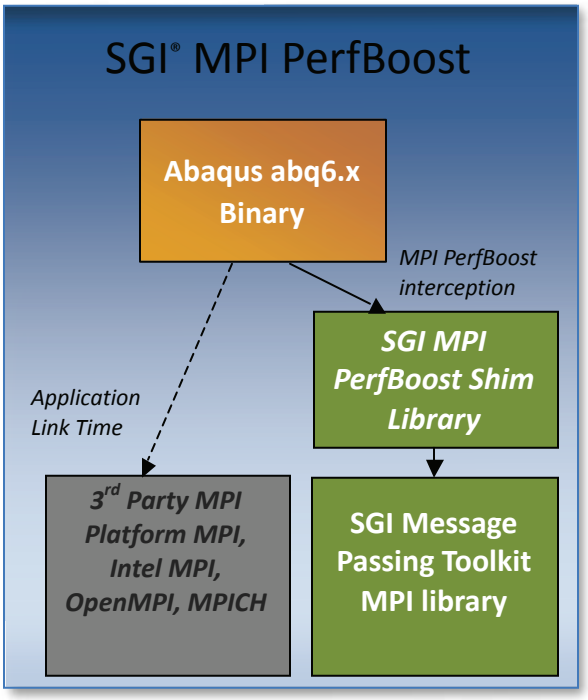



*Figure 10: SGI PerfBoost Messaging Flow to SGI MPI*

Figure 11 is an example of how SGI PerfBoost can provide the necessary hybrid MPI+OpenMP support to improve the efficiency of distribution of application threads to each of the requested cores. Within an SGI PerfBoost environment, we control the binding of MPI ranks and OpenMP threads based on policies. Using SGI PerfBoost can be 2x faster than using Platform MPI bundled with the Abaqus distribution. Platform MPI does not have strict process placement control as the SGI PerfBoost environment does on SMP systems.

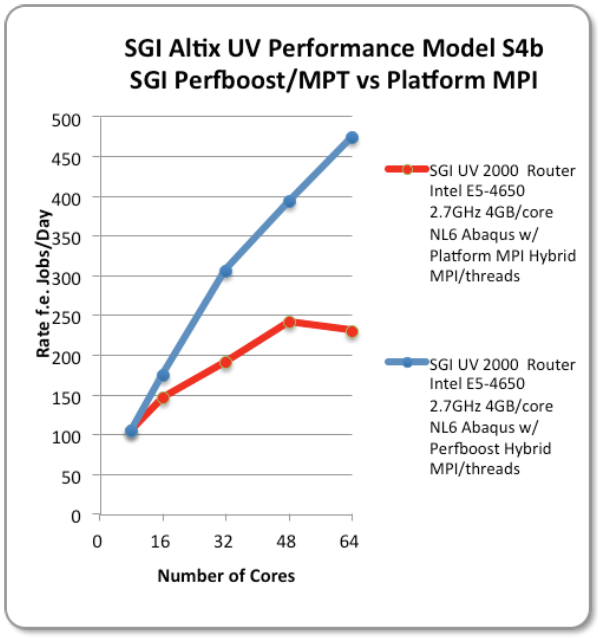



*Figure 11: SGI PerfBoost/MPI Performance*

The Abaqus start-up process executes a series of scripts to imitate the pre-processing stage where several python scripts are executed to define the Abaqus execution environment. The SGI Application Engineering team worked closely with the SIMULIA engineering group to develop a common global environment to allow preloading of the SGI PerfBoost library before the MPI environment is initialized. This environment allows SGI PerfBoost to intercept native MPI calls through Platform MPI and translates them into SGI MPI calls using a common global environment file. Using PerfBoost directives removes the user's interaction so they can focus on engineering analysis instead of setting up the custom PerfBoost environment file for each analysis.

Another compelling observation in favor of SGI PerfBoost is that using Platform MPI on a SMP system in conjunction with batch systems can lead to an unexpected thread process placement since Platform MPI doesn't have support for CPUSETs. CPUSETs[6] constrain the CPU and Memory placement of tasks to only the resources within a task's current CPUSET. CPUSETs provides an essential structure for managing dynamic job placement on large systems. Without CPUSETs, application threads are scattered throughout the SMP system passing control to the kernel scheduler which can lead to unpredictable runtimes.

There is also another interesting SGI MPI feature that can be realized on SMP systems. This feature is an MPI-2 extension and in particular the so-called one-sided MPI primitives (put and get). The use of one-sided primitives can lead to significant performance improvements due to considerably lower communication costs in comparison with traditional two-sided communication primitives (send, receive, etc). This improvement comes from two sources; significantly lower communication latencies and reduced number of synchronization barriers.

## 17.1 Effect of Hyper-Threading

When Intel first introduced the Intel® Xeon® 5500 series processor it included a new technology called Hyper-Threading (HT), HT, presents one core as two execution units as physical and logical cores. From an OS perspective when HT is enabled you will notice twice as many cores being available when querying /proc/cpuinfo output. Another tool which can be used on SGI platforms is the "cpumap" command. The "cpuset" command outputs processor information and the core IDs including the HT cores. Following is an example of the output from "cpumap" command in Figure 12.

```
r1i7n0 /store/sshaw> cpumap
Wed Jul 9 23:27:32 CDT 2014
r1i7n0.ice.americas.sgi.com

model name           : Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz
Architecture         : x86_64
cpu MHz              : 3000.000
cache size           : 25600 KB (Last Level)

Total Number of Sockets                : 2
Total Number of Cores                  : 20    (10 per socket)
Hyperthreading                         : ON
Total Number of Physical Processors    : 20
Total Number of Logical Processors     : 40    (2 per Phys Processor)

================================================================================

Processor Numbering on Node(s)

Node    (Logical) Processors
------    ------------------------
0       0   1   2   3   4   5   6   7   8   9  20  21  22  23  24  25  26  27  28  29
1      10  11  12  13  14  15  16  17  18  19  30  31  32  33  34  35  36  37  38  39
```

*Figure 12: The SGI cpumap command output*

To measure a performance gain with Hyper-Threading, we ran a few tests on a compute node with two Xeon® E5-2600 v2 series processors. The first test was 16 cores using just the physical cores on each processor with a runtime of 970 seconds. The second test was with 16 physical cores and 16 HT cores for a total of 32 cores analyzed. The runtime was 858 seconds. Rerunning the same 32 core analysis but with two 16 core compute nodes (without HT enable), the analysis completed in 582 seconds and is a 40% increase in performance over the 16 core single node test as shown in Figure 13.

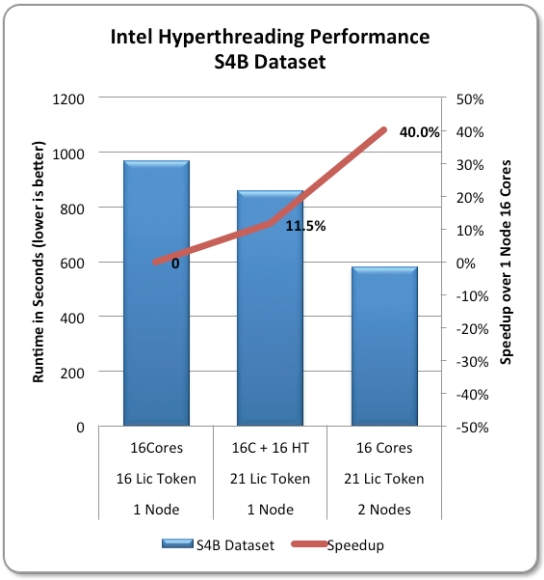



*Figure 13: Effects of Intel's Hyper-Threading Feature*

## 17.2 Effect of Core Frequency and Intel® Turbo Boost Technology

Turbo Boost is a feature, also first introduced in the Intel® Xeon® 5500 series, for increasing performance by raising the core operating frequency within controlled limits depending on the socket thermal envelope. The mode of activation is a function of how many cores are active at a given moment which may be the case when OpenMP threads or MPI processes are idle under their running parent. For example, for a base frequency of an Intel® E5-2667 v2 3.30GHz 8 core processor with one to two cores active will throttle up to ~4.0GHz, with three to four cores active only to ~3.80 GHz. The possible Turbo Boost frequencies based on active cores are shown below in Table 2. Figure 14 shows the Turbo Boost performance of active cores during an analysis with the S4B dataset using 16 cores per node.

| Max Core Frequency | E5-2667 3.30 GHz | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Turbo per Active Cores | 1c | 2c | 3c | 4c | 5c | 6c | 7c | 8c |
| Max Turbo Frequency (GHz) | 4.00 | 3.90 | 3.80 | 3.70 | 3.60 | 3.60 | 3.60 | 3.60 |

*Table 2: Turbo Boost Frequencies Based on Active Cores*

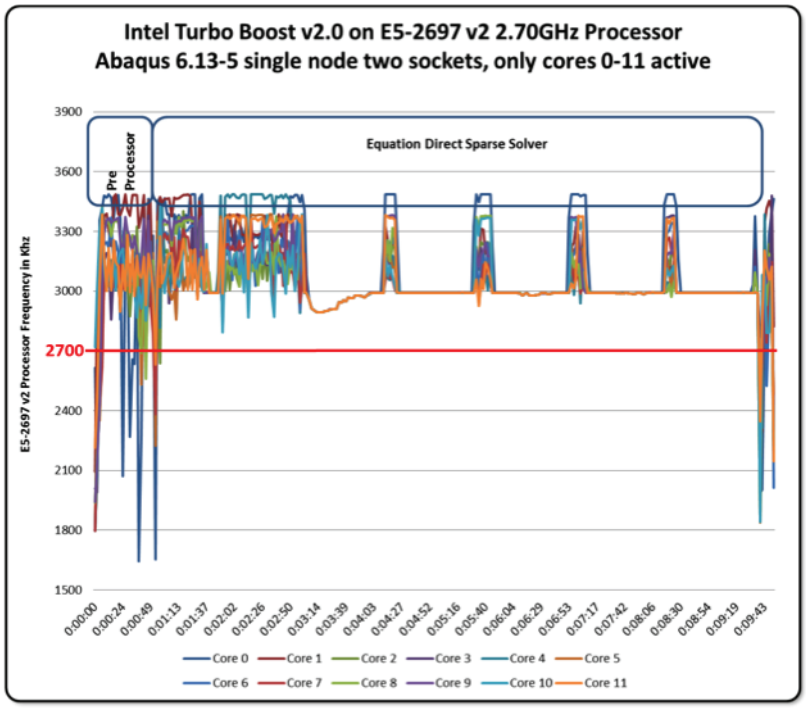



*Figure 14: Intel® Turbo Boost on E5 Series Processors*

For heavy computations, utilizing Intel's Turbo Boost technology can result in improved runtimes. The first generation of the Intel's Turbo Boost feature was limited to three frequency increments. With the new Intel® E5 series processors, the second generation of the Intel's Turbo Boost feature, the frequency increments can be eight 100MHz increments depending on the processor core activity. We recommend having Intel's Turbo Boost feature enabled in the system BIOS and Intel Speed Step Linux kernel modules loaded to fully benefit the performance from an Intel® E5 series processor.

When evaluating Intel® E5 series processors for Abaqus workloads, we find the price/performance sweet spot is the E5-2680 v2 2.80GHz processor for Rackable and ICE X clusters since it has low wattage and a Turbo Boost peak frequency of 3.60GHz. For SGI UV 2000 servers, we recommend Intel® Xeon® CPU E5-4627 v2 3.30 GHz processors since the Turbo Boost peak frequency is close to the performance of an E5-2680 v2 processor. Following is Table 3 with the various processor models, frequencies and wattage for comparison.

| | Intel® Ivy Bridge-EP & EX Processors | | | |
|---|---|---|---|---|
| **Model** | E5-2670 v2 (10c) | E5-2680 v2 (10c) | E5-2690 v2 (10c) | E5-4627 v2 (8c) |
| **Core Freq** | 2.50 GHz | 2.80 GHz | 2.90 GHz | 3.30 GHz |
| **Peak TB/1C Active** | 3.30 GHz | 3.60 GHz | 3.80 GHz | 3.80 GHz |
| **TB 8C Active** | 2.90 GHz | 3.10 GHz | 3.30 GHz | 3.60 GHz |
| **TDP** | 115W | 130W | 135W | 130W |

*Table 3: Intel® Processor Model, Frequencies and Wattage for Comparison*

# 18.0 Sizing Guidelines for Abaqus

When implementing an HPC platform for Abaqus each configuration can vary due to implicit and explicit workload combinations. Using some of the traditional scaling methods, you can breakdown a configuration into three types of groups based on degrees of freedom for implicit workloads or number of elements for explicit workloads. The three types of computing platforms are a small group cluster, medium departmental cluster or UV/SMP system or a large enterprise cluster or UV/SMP system. Following are some guidelines for implicit and explicit workloads. The charts shown below were derived from the SIMULIA Community Conference in 2008[7].

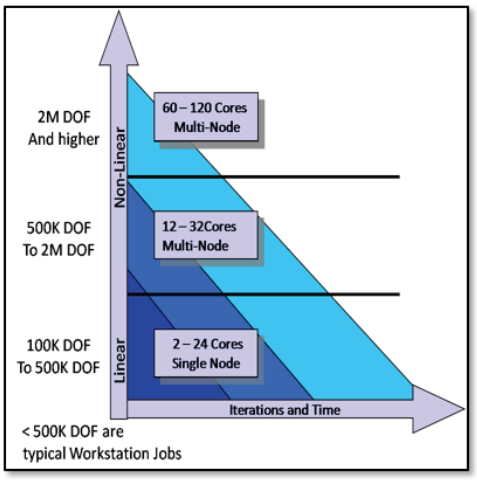
## 18.1 Implicit

**Linear Static Analysis**

- Linear jobs use SMP based eigensolvers which do not scale outside a single node. So in this case, an SGI UV or "fat" node with 12 or more cores will improve turnaround times.
- Extraction of natural frequencies and mode shapes require a high-performance storage file system for scratch data.
- Memory recommendation is 4-8GB/Core.

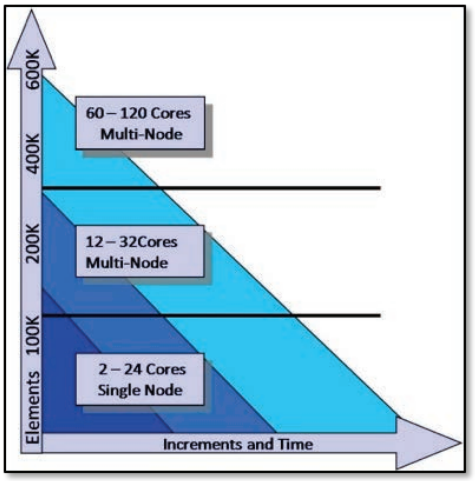
**Non-Linear Static Analysis**

- For small to medium-sized jobs of 500K to 2M DOF, we recommend 2-4GB Memory/Core.
- For large-sized jobs >2M DOF, the recommendation is 4-8GB Memory/Core.



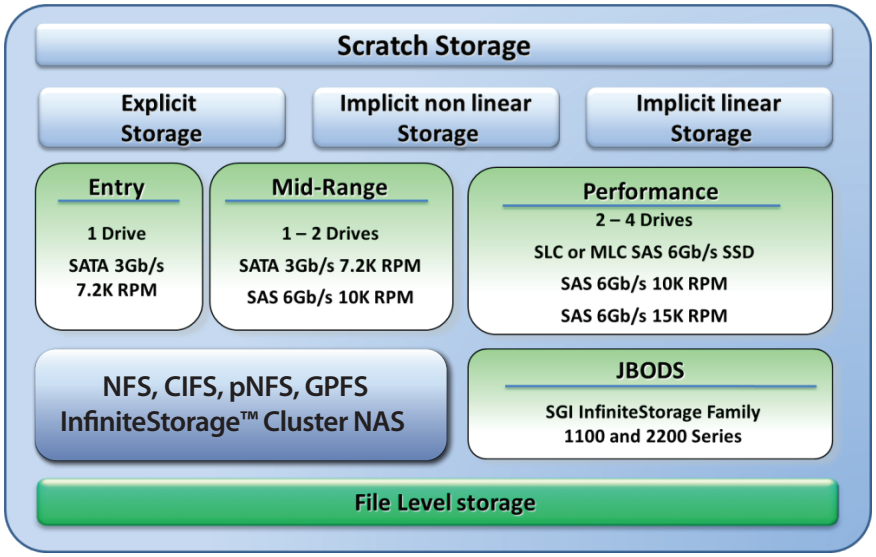

## 18.2 Explicit

- For small to medium-sized jobs of 100K to 400K elements, 2GB Memory/Core is common.
- For large jobs of 400K elements and above, 4GB of Memory/Core is recommended. Having more memory available allows the file I/O to be cached in the Linux buffer cache to allow better I/O performance.
- For complex model types, Abaqus/Explicit performance is impacted by high core counts due to messaging between nodes and scaling tapers at 64-128 cores.



## 18.3 Abaqus Storage Considerations

With computational structure mechanics, Abaqus implicit solvers can generate large result and restart files depending on the analysis and the frequency to output timestep information to the *.odb, *.stt, *.res and *.mdl files. As we mentioned previously in the Abaqus profiling section, we identified Abaqus I/O transfers to and from disk can be in the 1-8K range, and the analysis is dominated by write I/O operations. For implicit analysis with the direct sparse solver compared to analysis with the Lanczos or AMLS solvers, the I/O patterns can be very different. The Lanczos solver requires higher I/O demands due to the eigenvalue extraction of natural frequencies and frequent access to the disk subsystem. Below are some local scratch storage considerations based on solver and workflow.




# 19.0 About the SGI Systems

## 19.1 SGI Rackable® Cluster

SGI Rackable standard-depth, rackmount C2112-4RP4 servers support up to 512GB of memory per node in a dense architecture with up to 96 cores per 2U with support for up to 56 Gb/s. FDR and QDR InfiniBand, twelve-core Intel® Xeon® processor E5-2600 v2 series and DDR3 memory running SUSE® Linux® Enterprise Server or Red Hat® Enterprise Linux Server for a reduced TCO.

*Figure 15: Overhead View of Server with the Top Cover Removed and Actual Server*

## 19.2 SGI® ICE™ X System

The SGI ICE X is the world's fastest distributed memory supercomputer for over four years running. This performance leadership is proven in the lab, and at customer sites including the largest and fastest pure compute InfiniBand cluster in the world. The system can be configured with compute nodes comprising of Intel® Xeon® processor E5-2600 v2 series exclusively or with compute nodes comprising of both Intel® Xeon® processors and Intel® Xeon Phi™ coprocessors or Nvidia® compute GPU's. Running on SUSE® Linux® Enterprise Server and Red Hat® Enterprise Linux, SGI ICE X can deliver over 172 teraflops per rack and scale from 36 to tens of thousands of nodes.

SGI's ICE X can be architected in a variety of topologies with choice of switch and single or dual plane FDR InfiniBand interconnect topology. The integrated bladed design offers rack-level redundant power and cooling via air, warm water or cold water and is also available with storage and visualization options.

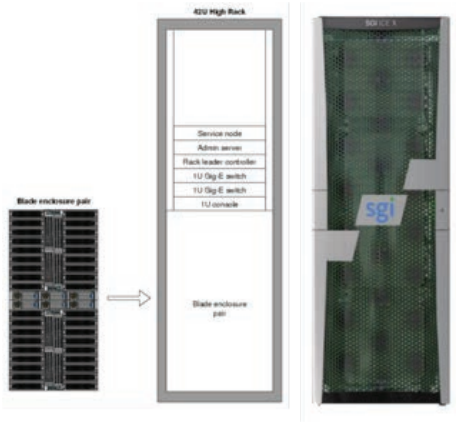



*Figure 16: SGI ICE X Cluster with Blade Enclosure Pair*

## 19.3 SGI® UV™ 2000

The SGI UV 2000 is a scalable cache-coherent shared memory architecture. SGI's UV 2 product family can scale a single system image (SSI) to a maximum of 2,048 cores (4,096 threads) due to its NUMAflex®, blade-based architecture. SGI UV 2 includes the Intel® Xeon® processor E5-4600 and the latest Intel® Xeon® processor E5-4600 v2 product family. This system can operate unmodified versions of Linux such as SUSE Linux Enterprise Server and Red Hat® Enterprise Linux. The SGI UV also supports scalable graphics accelerator cards, including NVIDIA® Quadro®, NVIDIA® Tesla® K40 GPU computing accelerator and Intel® Xeon Phi™. The memory is allocated independently from core allocation for maximum multi-user, heterogeneous workload environment flexibility. Whereas on a cluster, problems have to be decomposed and require many nodes to be available, the SGI UV can run a large memory problem on any number of cores and application license availability with less concern of the job getting killed for lack of memory resources compared to limitations of a cluster platform.

*Figure 17: SGI UV 2000 SMP and Compute and Base I/O blades*

## 19.4 SGI Performance Tools

SGI Performance Suite takes Linux performance software to the next level. While hardware and processor technology continue to scale, managing software performance has become increasingly complex. SGI continues to extend technical computing performance for large scale servers and clusters. SGI Performance Suite incorporates the most powerful features and functionality from SGI ProPack™ 7, combined with several new tools and enhancements, and new, more flexible product packaging which allows you to purchase only the component or components that you need.


*Figure 18: SGI Performance Suite Components*

## 19.5 SGI System Management Tools

SGI Management Center provides a powerful yet flexible interface through which to initiate management actions and monitor essential system metrics for all SGI systems. It reduces the time and resources spent administering systems by improving software maintenance procedures and automating repetitive tasks ultimately lowering the total cost of ownership, increasing productivity, and providing a better return on the customer's technology investment. SGI Management Center is available in multiple editions which tailor features and capabilities to the needs of different administrators, and makes available optional features that further extend system management capabilities.

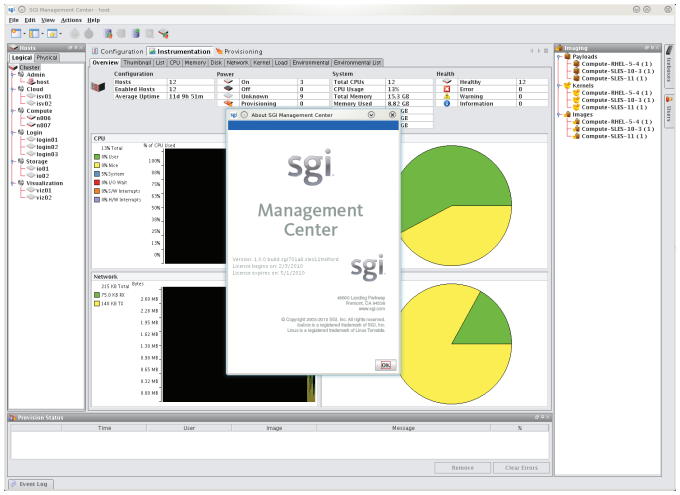



*Figure 19: SGI Management Center Web Interface*

## 19.6 Resource & Workload Scheduling

Resource and workload scheduling allows you to manage large, complex applications, dynamic and unpredictable workloads, and optimize limited computing resources. SGI offers several solutions that our customers can choose from to best meet their needs.

**Altair Engineering PBS Professional®**

Altair PBS Professional® is SGI's preferred workload management tool for technical computing scaling across SGI's clusters and servers. PBS Professional is sold by SGI and supported by Altair Engineering and SGI.

**Features:**

- Policy-driven workload management which improves productivity, meets service levels, and minimizes hardware and software costs
- Integrated operation with SGI Management Center for features such as workload driven, automated dynamic provisioning

**Adaptive Computing Moab® HPC Suite Basic Edition**

Adaptive Computing Moab® HPC Suite enables intelligent predictive scheduling for workloads on scalable systems.

**Features:**

- Policy-based HPC workload manager that integrates scheduling, managing, monitoring and reporting of cluster workloads
- Includes TORQUE resource manager

# 20.0 Summary

In this paper, we presented and analyzed the performance of Abaqus/Standard and Abaqus/Explicit datasets on three SGI platforms including Nvidia® Tesla™ K40 GPUs. Based on the three datasets used, we also identified compute, memory bandwidth and communication bound types of problems using analysis techniques we commonly use when performing benchmarks.

Listening to and working with our CAE customers for many years, SGI delivers a unified compute and storage solution that reduces overall system management requirements and cost as well as simplified data management and archival needs. SGI's flexible x86 based server portfolio can scale effortlessly to meet your compute and I/O requirements as they evolve. The SGI UV product family provides the only single node, cache-coherent shared memory platform that can start small and grow seamlessly as your needs develop. Built with Intel® Xeon® processors E5 family, SGI's UV 2000 is capable of consolidating the entire CAE workflow including Abaqus/Standard and Abaqus/Explicit onto a single platform. The SGI ICE X and SGI Rackable servers provide a best of breed factory integrated and tested cluster computing environment. And finally, SGI Modular InfiniteStorage provides the ability to store and access the vast amount of engineering data created by these CAE bandwidth-intensive applications.

## 21.0 References

[1] Intel Corporation, Intel® AVX Optimization on 2nd generation Intel® Core™ Processors, May 9th, 2011

[2] Intel Corporation, www.intel.com Intel® Xeon® Processor E5 Family

[3] Intel Corporation, www.intel.com Intel® Xeon® Processor X5600 Family

[4] Oak Ridge National Lab, Future Technologies Group, https://github.com/spaffy/shoc/wiki

[5] D. Thomas, J.P Panziera, J. Baron, MPInside: a Performance Analysis and Diagnostic Tool for MPI Applications, Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering, 79-86, 2010.

[6] CPUSET, Linux Programmer's Manual, www.kernel.org

[7] 2008 SIMULIA Community Conference proceedings

## 22.0 About SGI

SGI is a global leader in high performance solutions for compute, data analytics and data management that enable customers to accelerate time to discovery, innovation, and profitability. Visit sgi.com for more information.

## 23.0 About Dassault Systèmes SIMULIA

Dassault Systèmes SIMULIA applications, including Abaqus, fe-safe, Isight, Tosca, and Simulation Lifecycle Management, enable users to leverage physics-based simulation and high-performance computing to explore real-world behavior of products, nature, and life. As an integral part of Dassault Systèmes **3D**EXPERIENCE® platform, SIMULIA realistic simulation applications accelerate the process of making highly-informed, mission-critical design and engineering decisions, before committing to costly and time-consuming physical prototypes. Visit "http://www.3ds.com/simulia" for more information.

**Global Sales and Support**: sgi.com/global