TECH GUIDE

SGI® Technology Guide for ANSYS® Mechanical Analysts

September, 2015

Author

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

Abstract

ANSYS Mechanical tackles all important Normal Mode Analysis utilizing either Shared Memory Parallelism (SMP) or Distributed Memory Parallelism (DMP). The technical approaches used here include block lanczos and supernode eigensolver. Efficient execution of the above two paradigms and solutions requires extreme care in allocating hardware resources. The topic is even more important given hardware variety today, ranging from single node multi-core workstations through multiple node clusters to single image many-core systems addressing very large memory space. The paper will explore trade-offs in computer resource requirements of such solutions to establish guidelines for advanced SGI computer hardware systems.

† Senior SGI Applications Engineer
†† Director of SGI Virtual Product Development Solutions
††† Principle SGI Applications Engineer

TABLE OF CONTENTS

1.0 About SGI Systems

SGI systems used to perform the benchmarks outlined in this paper include the SGI Rackable® standard depth cluster; SGI® ICE™ XA integrated blade cluster and the SGI® UV™ 2000 shared memory system. They are the same servers used to solve some of the world's most difficult computing challenges.

1.1 SGI Rackable® Standard-Depth Cluster

SGI Rackable standard-depth, rackmount C2112-4GP3 2U enclosure supports four nodes and up to 4TB of memory in 64 slots (16 slots per server). It also supports up to 144 cores per 2U with support of FDR InfiniBand, fourteen-core Intel® Xeon® processor E5-2600 v3 series and 2133 MHz DDR4 memory running SUSE® Linux® Enterprise Server or Red Hat® Enterprise Linux for a reduced TCO (Figure 1).

SGI Rackable cluster configuration used in this paper:

Benchmark System

- Nodes: 64
- Processor: Intel® Xeon® 14-core @ 2.60GHz (E5-2697 v3)
- IB: FDR Interconnect (Mellanox® OFED: 2.4-1.0.0)
- Total Memory Per Node: 128 GB (4.6 GB per core); Memory Speed: 2133 MHz
- OS: SUSE Linux Enterprise Server 11 sp3
- Altair® PBS Professional Batch Scheduler
- SGI Software: SGI MPI; SGI Performance Suite

Figure 1: *Overhead View of SGI Rackable Server with the Top Cover Removed*

1.2 SGI® ICE™ XA System

SGI® ICE™ XA is one of the world's fastest commercial distributed memory supercomputer. 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 Intel® Xeon® processor E5-2600 v3 series exclusively or with compute nodes comprising 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 XA can deliver over 191 teraflops of performance per rack and scale from 36 to tens of thousands of nodes.

SGI ICE XA is designed to minimize system overhead and communication bottlenecks and can be architected in a variety of topologies with choice of switch and single or dual plane FDR interconnect. The integrated bladed design offers rack-level redundant power and cooling via air (currently ICE X provides air cooled racks not ICE XA), warm or cold water and is also available with storage and visualization options (Figure 2).

SGI ICE XA configuration used in this paper:

- Nodes: 576
- Processor: Intel® Xeon® 12-core @ 2.60GHz (E5-2690 v3)
- IB: FDR Interconnect (Mellanox® OFED: 2.4-1.0.0)
- Total Memory Per Node: 128 GB (5.33 GB per core); Memory Speed: 2133 MHz
- OS: SUSE Linux Enterprise Server 11 sp3
- Altair® PBS Professional Batch Scheduler
- SGI Software: SGI MPI; SGI Performance Suite

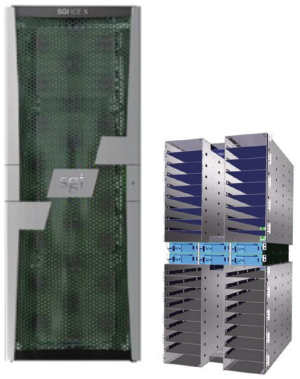

Figure 2: *SGI ICE X Cluster with Blade Enclosure*

1.3 SGI® UV™ 2000

SGI UV 2000 server comprises up to 256 sockets (2,048 cores). Support for 64TB of global shared memory in a single system image enables efficiency of SGI UV for applications ranging from in-memory databases, to diverse sets of data and compute-intensive HPC applications all the while programming via the familiar Linux OS [2], without the need for rewriting software to include complex communication algorithms. TCO is lower due to one-system administration needs. Workflow and overall time to solution is accelerated by running Pre/Post-Processing, solvers and visualization on one system without having to move data (Figure 3).

Job memory is allocated independently from cores 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 adapting to application license availability and with less concern about lack of memory resources killing the job.

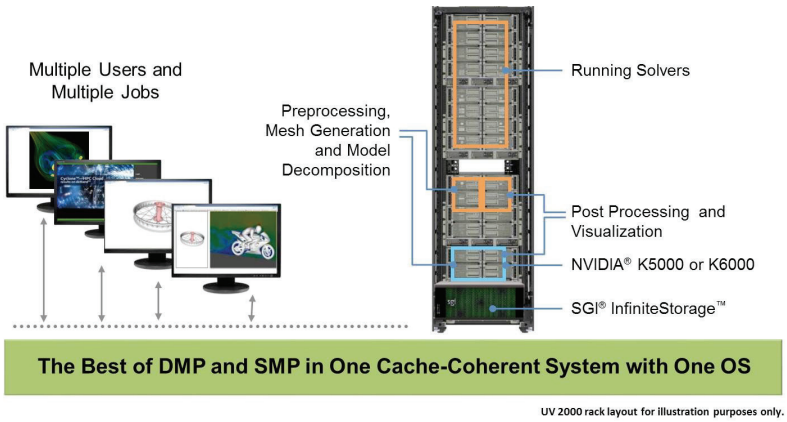

Figure 3: *SGI UV CAE workflow running ANSYS applications*

SGI UV shared memory configurations used in this paper:

- Nodes: 1 (512 cores/ 64 Sockets)
- Processor: Intel® Xeon® 8-core @ 3.30GHz (E5-4627 v2) and 10-core 2.40GHz (E5-4650v2)
- Interconnect: SGI NUMALink® 6, Quad-Plane Routered Fat Tree
- Total Memory: 4TB; Speed: 1867 MHz
- OS: SUSE Linux Enterprise Server 11 sp3
- Altair® PBS Professional Batch Scheduler
- SGI Software: SGI MPI; SGI Performance Suite

1.4 SGI Performance Tools

Utilizing the latest MPI compliant libraries and standard-distribution Linux, SGI® Performance Suite (Figure 4) fuels HPC applications to achieve breakthrough speed and scale. A feature-rich tool set optimizes application placement, enables application tuning at runtime without recompiling, and can boost performance up to 70%. Fine-grain metrics facilitate MPI analysis. Checkpoint Restart augments productivity. And hard real-time performance can be realized without special kernels on standard Linux. Coupled with world-class application expertise, SGI takes Linux to the next level. For detailed information: http://www.sgi.com/products/software/.

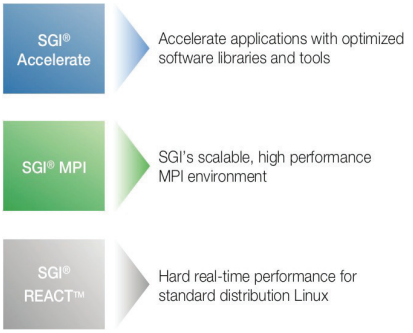

Figure 4: *SGI Performance Suite Component*

1.5 SGI System Management Tools

Spanning bare-metal provisioning and protection against memory failure, to 24x7 systems monitoring, task automation, and innovative power optimization, SGI® Management Suite helps maximize productivity and achieve a high return on your investment. Administrators can deploy systems and upgrades with unparalleled speed, proactively manage system health and energy consumption, and deliver consistently high service levels— enabling users to run more jobs in less time and without interruption. For detailed information: http://www.sgi.com/products/software/smc.html

1.6 Resource and Workload Scheduling

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

Altair Engineering PBS Professional® is SGI's preferred workload management tool for technical computing scaling across SGI's clusters and servers. 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
- Altair PBS Professional Power Awareness integrates job-level power management with SGI Management Center 3

Adaptive Computing Moab® HPC Suite Basic Edition

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

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

1.7 SGI VizServer® with NICE DCV

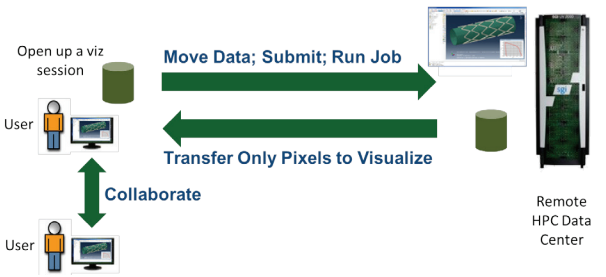

Figure 5: SGI VizServer workflow

SGI VizServer with NICE DCV installed on a company's servers can provide remote visualization capabilities through a software-as-a-service (SaaS) built in the company's private network. The Pre/Post software is accessed through an easy-to-use web interface, resulting in simplicity for the end user. This solution provides intuitive help and guidance to ensure that less-experienced users can maximize productivity without being hindered by complex IT processes.

SGI VizServer with NICE DCV Components:

- Engineer-friendly self-service portal: The self-service portal enables engineers to access their Pre/Post application and data in a web browser–based setting. It also provides security, monitoring, and management to ensure that users cannot leak company data and that IT managers can track usage. Engineers access their Pre/Post application and data directly from their web browsers, with no need for a separate Prep/Post software installation on their local client.
- Resource control and abstraction layer: The resource control and abstraction layer lies underneath the portal, not visible to end users. It handles job scheduling, remote visualization, resource provisioning, interactive workloads, and distributed data management without detracting from the user experience. This layer translates the user request from the browser and facilitates the delivery of resources needed to complete the visualization or HPC tasks. This layer has a scalable architecture to work on a single SGI Rackable cluster or SGI UV server, as well as a multi-site WAN implementation.
- Computational and storage resources: The SGI VizServer with NICE DCV software takes advantage of the company's existing or newly purchased SGI industry-standard resources, such as servers, HPC schedulers, memory, graphical processing units (GPUs), and visualization servers, as well as the required storage to host application binaries, models and intermediate results. These are all accessed through the web-based portal via the resource control and abstraction layer and are provisioned according to the end user's needs by the middle software.

The NICE DCV and EnginFrame software is built on common technology standards. The software adapts to network infrastructures so that an enterprise can create its own secure engineering cloud without major network upgrades. The software also secures data, removing the need to transfer it and stage it on the workstation, since both technical applications and data stay in the private cloud or data center. These solutions feature the best characteristics of cloud computing—simple, self-service, dynamic, and scalable, while still being powerful enough to provide 3D visualization as well as HPC capabilities to end users, regardless of their location.

2.0 ANSYS Mechanical Overview

ANSYS Mechanical solves structural linear or nonlinear and dynamics analysis problems using a library of equation solvers comprising of a sparse direct solver, preconditioned conjugate gradient (PCG) iterative solver and a Jacobi conjugate gradient (JCG) solution.

2.1 Parallel Processing Capabilities of ANSYS Mechanical

ANSYS Mechanical solvers are parallel processing capable using Shared Memory Parallel (SMP) and Distributed Memory Parallel (DMP) modes [1]. The implementation and application on SGI systems will be developed in the following sections.

2.1.1 Underlying Hardware and Software Notions

It is important to distinguish hardware components of a system and the actual computations being performed using them. On the hardware side, one can identify:

1. Cores, the Central Processing Units (CPU) capable of arithmetic operations.
2. Processors, the four, six, eight and up core socket-mounted devices.
3. Nodes, the hosts associated with one network interface and address.

With current technology, nodes are implemented on boards in a chassis or blade rack-mounted enclosure. The board may comprise two sockets or more.

From the software side, one can identify:

1. Processes: execution streams having their own address space.
2. Threads: execution streams sharing address space with other threads.

Therefore, it is important to note that processes and threads created to compute a solution on a system will be deployed in different ways on the underlying nodes through the processors and cores' hardware hierarchy.

2.1.2 Parallelism Background

Parallelism in scientific/technical computing exists in two paradigms implemented separately but sometimes combined in '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 relative importance of arithmetic operations versus data access referred to as 'DO loop granularity.' In the late 1990's, Distributed Memory Parallelism (DMP) Processing was introduced and proved very suitable for performance gains because of its coarser grain parallelism design. It consolidated on the MPI Application Programming Interface. In the meantime, Shared Memory Parallelism saw adjunction of mathematical libraries already parallelized using efficient implementation through OpenMP™ (Open Multi-Processing) and Pthreads standard API's.

Both DMP and SMP (with some limitations) programs can be run on the two commonly available types of hardware systems:

- Shared Memory systems or single nodes with multiple cores sharing a single memory address space and a single instance of the operating system.
- Distributed Memory systems, otherwise known as clusters, comprised of nodes with separate local memory address spaces and a dedicated instance of the operating system per node.

Note: SMP programs because of their single memory spaces cannot execute across clusters. Inversely, DMP programs can run perfectly well on a Shared Memory system. Since DMP has coarser granularity than SMP, it is therefore preferable, on a Shared Memory system, to run DMP rather than SMP despite what the names may imply at first glance. SMP and DMP processing may be combined together, in what is called 'hybrid mode'.

2.1.3 Distributed Memory Parallel Implementations

Distributed Memory Parallelism is implemented through the problem at hand with domain decomposition. Depending on the physics involved in their respective industry, the domains could be geometry, finite elements, matrix, frequency, load cases or right hand side of an implicit method. Parallel inefficiency from communication costs is affected by the boundaries created by the partitioning. Load balancing is also important so that all MPI processes perform the same number of computations during the solution and therefore finish at the same time. Deployment of the MPI processes across the computing resources can be adapted to each architecture with 'rank' or 'round-robin' allocation.

2.1.4 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):

Amdahl Speedup=1/[(1-P)+P/N]

A derived metric is: Efficiency=Amdahl Speedup/N

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

N	System a elapsed seconds	System B elapsed seconds
1	1000	810
10	100	90
Speedup	10	9

System A and System B could show parallel speedups of 10 and 9, respectively, even though System B has faster raw performance across the board. Normalizing speedups with the slowest system serial time remedies this problem:

Speedup	10	11.11

A computational solution of a particular dataset is said to exhibit strong scalability if elapsed execution time decreases when number of processors increases. While computational solution of increasing dataset sizes is said to exhibit weak scalability when elapsed execution time can remain bounded through an increase of number of processors.

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

Number of jobs/hour/system = 3600/(Job elapsed time)

The system could be a chassis, rack, blade, or any hardware provisioned as a whole unit.

2.2 Parallel Execution Control

2.2.1 Submittal Procedure

Submittal procedure must ensure:

1. Placement of processes and threads across nodes and also sockets within nodes.
2. Control of process memory allocation to stay within node capacity.
3. Use of adequate scratch files across nodes or network.

Batch schedulers/resource managers dispatch jobs from a front-end login node to be executed on one or more compute nodes so the following is a possible synoptic of a job submission script:

1. Change directory to the local scratch directory on the file compute node allocated by the batch scheduler.
2. Copy all input files over to this directory.
3. Create parallel local scratch directories on the other compute nodes allocated by the batch scheduler.
4. Launch application on the first compute node. The executable may itself carry out propagation and collection of various files between launch node and the others at start, and end of the main analysis execution. The launch script may also asynchronously sweep up output files to free up scratch directory.

2.2.2 Submittal Command

The following keywords are used for the ANSYS execution command:

- -dis: Enables Distributed Memory Parallelism.
- -np argument: Number of distributed processes (DMP) in the solution and number of Open Multi-Processing (OpenMP) threads during all processing that has been parallelized.

2.2.3 Software

- ANSYS Mechanical Release R16.1
- Compilers: Fortran: Intel® Fortran Compiler 11.1 for EM64T-based applications
- MPI: IBM® Platform MPI, Intel® MPI, SGI® MPI

2.3 Tuning

2.3.1 Input/Output and Memory

To achieve the best runtime in a batch environment, disk access to input and output files should be placed on the high performance filesystem closest to the compute node. The high performance filesystem could be either an in-memory filesystem (/dev/shm), a Direct (DAS) or a Network (NAS) Attached Storage filesystem. In diskless computing environments, in-memory filesystem or Network Attached Storage are the only options. In cluster computing environments with a Network Attached Filesystem (NAS), isolating application MPI communications and NFS traffic will provide the best NFS I/O throughput for scratch files. The filesystem nomenclature is illustrated in Figure 6.

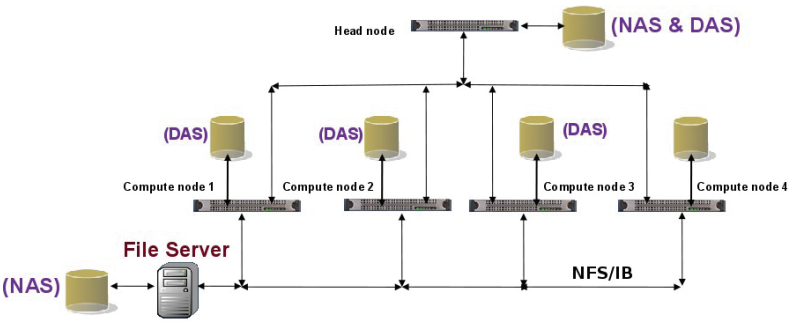

Figure 6: *Example File Systems for Scratch Space*

Having more memory per core will increase performance since it can be allocated for the analysis as well as the Linux kernel buffer cache to improve I/O efficiency. SGI's Flexible File I/O (FFIO) is a link-less library (which means it does not need to be linked to the application) bundled with SGI Accelerate. It implements user defined I/O buffer caches to avoid the operating system buffer caches from thrashing when running multiple I/O intensive jobs or processes. This can be effective in Shared Memory Parallel systems or cluster computing environments using DAS or NAS storage subsystems. FFIO isolates user page caches so jobs or processes do not contend for Linux Kernel page cache. Hence, FFIO minimizes the number of system calls and I/O operations as echoed back by the eie_close sync and async values reflecting synchronous calls to disk—which should be as close to 0 as possible—to and from the storage subsystem and improves performance for large and I/O intensive jobs. (Ref [1], Chapter 7 Flexible File I/O).

2.3.2 Using Only a Subset of Available Cores on Dense Processors

Two ways of looking at computing systems are either through nodes which are their procurement cost sizing blocks or through cores which are their throughput sizing factors. When choosing metrics, 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 so this may provide performance increase possibilities. The deployment of threads or processes across partially used nodes should be done with consideration to the existence of shared resources among cores.

2.3.3 Hyper-Threading

Intel® Hyper-threading (HT) is a feature of the Intel® Xeon® processor family which can increase performance for multi-threaded or multi-process applications. It allows a user to run twice the number of OpenMP threads or MPI processes than available physical cores per node (over-subscription).

Note, beyond 2 nodes, with ANSYS Mechanical, Hyper-threading gains are negated by added communication costs between the double-up numbers of MPI processes.

2.3.4 Intel® Turbo Boost

Intel® Turbo Boost is a feature of the Intel® Xeon® processor family, for increasing performance by raising the core operating frequency within controlled limits constrained by the thermal envelope of the processor. The mode of activation is a function of how many cores are active at a given moment when MPI processes, OpenMP threads or Pthreads are running. Turbo Boost can improve performance for low numbers of cores used, up to the ratio of the maximum frequency over baseline value. As more cores are used, Turbo Boost cannot increase the frequencies on all of them as it can on fewer active ones. For example, for a base frequency of 3.0GHz, when 1-2 cores are active, core frequencies might be throttled up to 3.3GHz, but with 3-4 cores active, frequencies may be throttled up only to 3.2 GHz. For computational tasks, utilizing Turbo Boost often results in improved runtimes so it is best to leave it enabled, although the overall benefit may be mitigated by the presence of other performance bottlenecks outside of the processor.

2.3.5 SGI Performance Suite MPI and SGI PerfBoost

The ability to bind an MPI rank to a processor core is key to control performance on the multiple node/socket/core environments available. From [3], '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 its 'omplace' option enforces accurate placement of Hybrid MPI processes, OpenMP threads and Pthreads within each node. SGI MPI's bundled PerfBoost facility linklessly translates IBM Platform MPI, Intel® MPI, OpenMPI calls on the fly to SGI MPI calls.

2.3.6 SGI Accelerate LibFFIO

ANSYS/Mechanical/APDL is not very I/O intensive and placement can be handled by SGI MPI, therefore, libFFIO is not often necessary. However, for larger I/O's libFFIO can compensate for bandwidth contention on NAS or slow filesystems.

3.0 Results

This section describes the ANSYS Mechanical benchmarks models used and the results.

3.1 Benchmark Examples

Job	Description	GPU	Notes
V16cg-1	JCG solver, symmetric matrix, 5300k DOFS, static, linear, thermal analysis	NVIDIA[1]	Medium sized job for iterative solvers, good test of memory bandwidth
V16cg-2	PCG solver, symmetric matrix, 12300k DOFs, static, linear, structural analysis	NVIDIA[1]	Large sized job for iterative solvers, good test of memory bandwidth
V16cg-3	PCG solver, symmetric matrix, 14200k DOFs, static, linear, structural analysis	None	Large sized job for iterative solvers, good test of processor flop speed and memory bandwidth
V16ln-1	PCG Lanczos eigensolver, symmetric matrix, 7700k DOFs, modal, linear, structural analysis requesting 10 modes	NVIDIA[1]	Medium sized job for iterative solvers, good test of memory bandwidth
V16ln-2	Subspace eigensolver, symmetric matrix, 2000k DOFs, modal, cyclic symmetry, linear, structural analysis requesting 50 modes	NVIDIA, Intel[2]	Medium sized job for direct solvers, should run incore on machines with 64 GB or more of memory, good test of processor flop speed and memory bandwidth
V16sp-1	Sparse solver, non-symmetric matrix, 650k DOFs, static, nonlinear, thermal-electric coupled field analysis	NVIDIA	Medium sized job for direct solvers, should run incore on machines with 32 GB or more of memory, good test of processor flop speed if running incore and I/O if running out-of-core
V16sp-2	Sparse solver, symmetric matrix, 4700k DOFs, transient, nonlinear, structural analysis	NVIDIA, Intel[2]	Medium sized job for direct solvers, should run incore on machines with 48 GB or more of memory, good test of processor flop speed if running incore and I/O if running out-of-core
V16sp-3	Sparse solver, symmetric matrix, 1700k DOFs, harmonic, linear, structural analysis requesting 1 frequency.	NVIDIA, Intel[2]	Medium sized job for direct solvers, should run incore on machines with 64 GB or more of memory, good test of processor flop speed if running incore and I/O if running out-of-core
V16sp-4	Sparse solver, symmetric matrix, 3200k DOFs, static, nonlinear, structural analysis with 1 iteration	NVIDIA, Intel[2]	Large sized job for direct solvers), should run incore on machines with 96 GB or more of memory, good test of processor flop speed if running incore and I/O if running out-of-core
V16sp-5	Sparse solver, symmetric matrix, 6000k DOFs, transient, nonlinear, structural analysis with 1 iteration	NVIDIA, Intel[2]	Large sized job for direct solvers, should run incore on machines with 128 GB or more of memory, good test of processor flop speed if running incore and I/O if running out-of-core

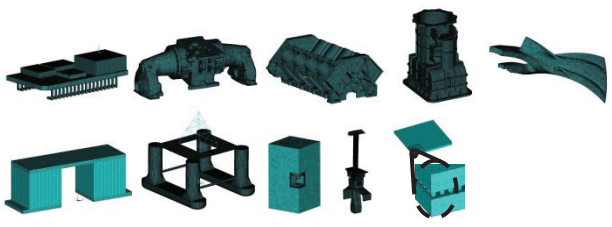

Figure 7: ANSYS 16.0 Hardware Benchmark Images

3.2 Benchmark Results

Two types of charts have been used traditionally. The file type highlights the slower end of the performance curve by plotting elapsed times in seconds versus the number of threads, processes, cores or nodes. The second type used in this paper is more useful for the higher end of the performance curve by plotting the number of jobs per day (Y axis, higher is better) versus the number of the same resource units mentioned above brought about to decrease turnaround time.

3.2.1 Optimal Serial Runs

Figure 8, 'Speed gains on single core' show how single core runs benefit from Turbo Boost on a Intel® Xeon® processor. DMP mode appears less efficient than SMP mode for 1 core runs because of the overhead entailed by DMP.

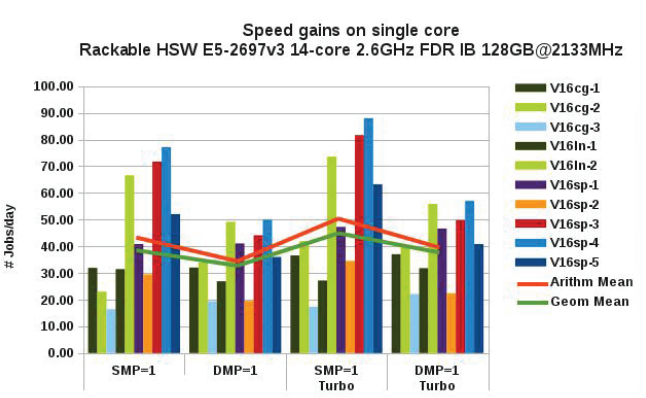

Figure 8

3.2.2 Scaling, Parallel Runs on Single Nodes

Figure 9 'Speed gains on single 28-core node' shows for the Intel® Xeon® E5-2697v3 processors how DMP processing improved performance from single core processing more than SMP processing did. Turbo Boost is beneficial as shown in the fourth group of runs. Hyper-threading decreases performance as shown in groups of runs where 56 SMP threads or MPI processes oversubscribe 28 physical cores. Therefore hyper-threading cannot be recommended.

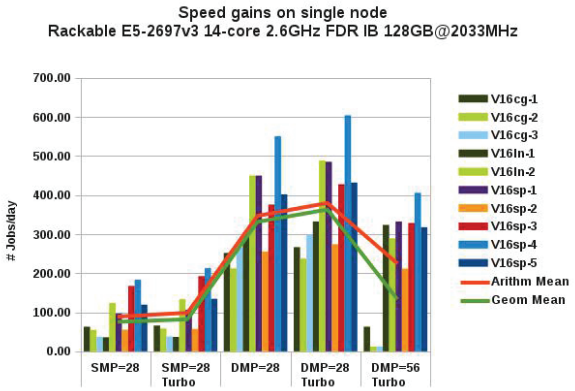

Figure 9

3.2.3 Frequency Scaling, Parallel Runs on Single Nodes

Figures 10 and 11: Frequency comparison on one 24 core ICE X node illustrate that increase in frequency (2.6/2.0=1.3) does not translate fully into the same performance increase (1.18).

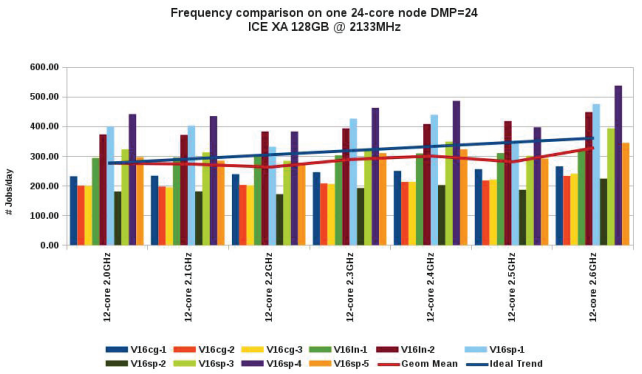

Figure 10

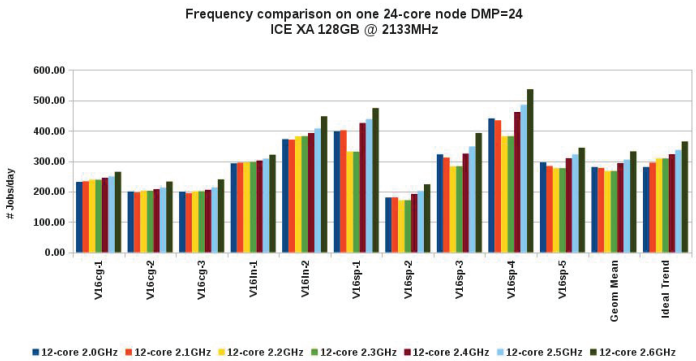

Figure 11

3.2.4 Scaling of Parallel Runs on Multiple Nodes

Figure 12: Speed gains on multiple 28-fully-utilized-cores show higher performance can be achieved by going from 1 to 2 and 3 nodes as denoted by the right-hand side legend.

On the other hand, when the interconnect is GigE, there is no scaling past 1 node as shown by Figure 13: Speed losses on multiple 28-core nodes.

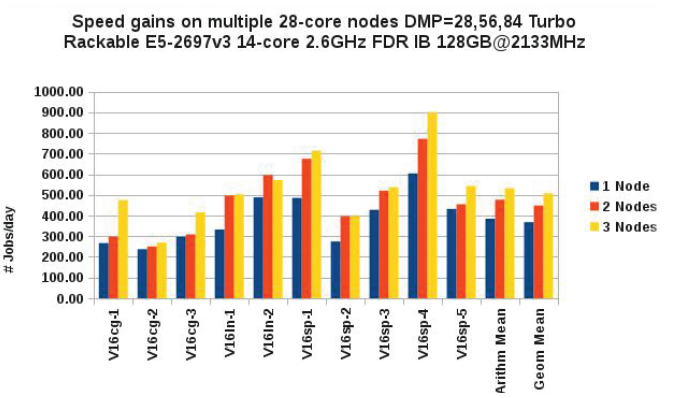

Figure 12

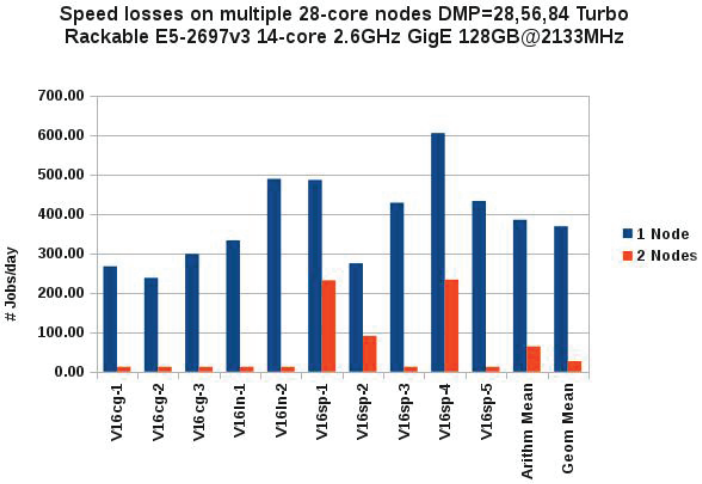

Figure 13

3.2.5 MPI Library Effect on Multiple Node Scaling

Figure 14, 15 and 16 show that on average, SGI MPT performs better on the standard suite of benchmarks for 1, 2 and 3, 28-fully-utilized-core nodes.

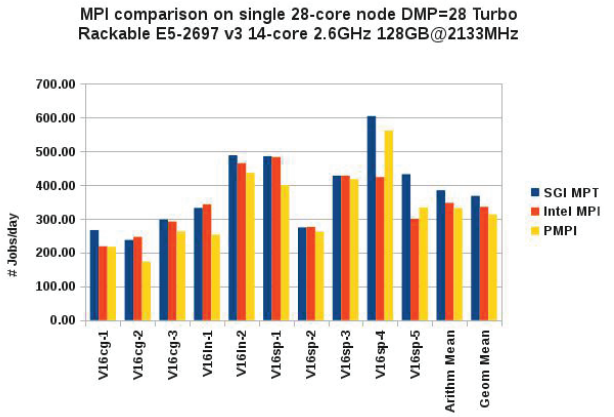

Figure 14

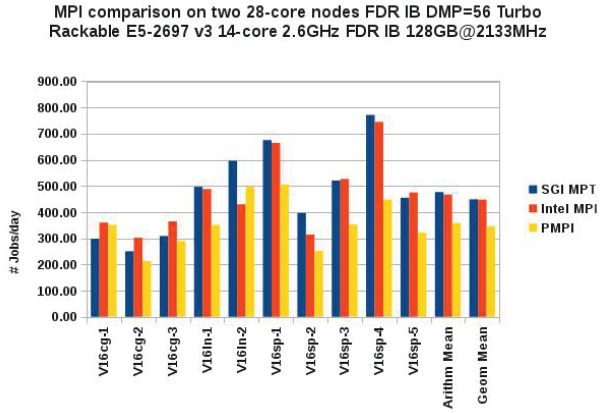

Figure 15

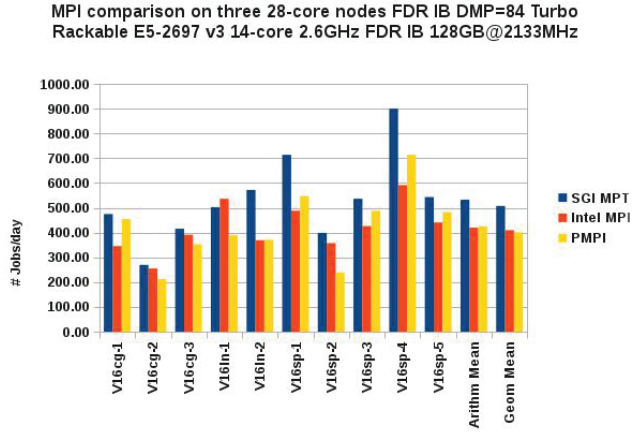

Figure 16

3.2.6 More Cores or Higher Frequency?

Figure 17 shows more cores are better at the same frequency but when it is markedly lower as on Figure 18 then fewer cores can be better.

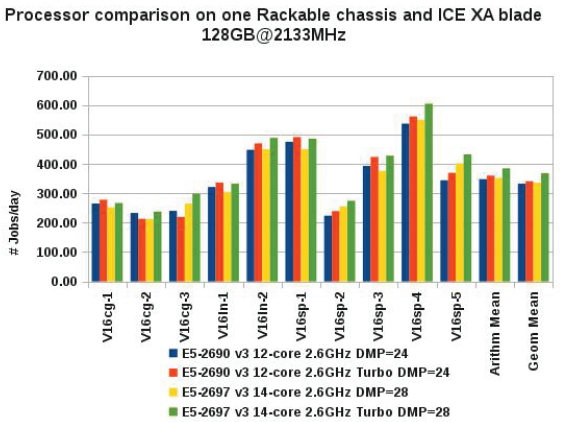

Figure 17

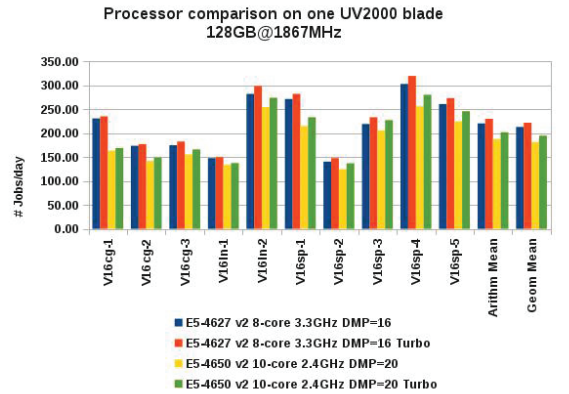

Figure 18

3.2.7 SGI UV 2000 Scaling

Figures 19 and 20 and show how, on a UV 2000, turnaround time can be reduced by allocating more cores to running jobs on either Intel® Xeon® 8-core E5-4627v2 or Intel® Xeon® 10-core E5-4650v2.

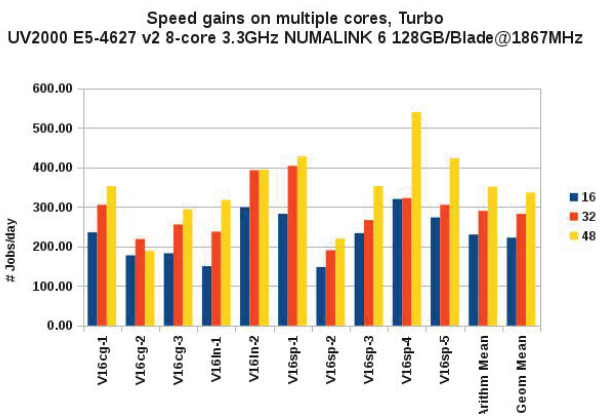

Figure 19

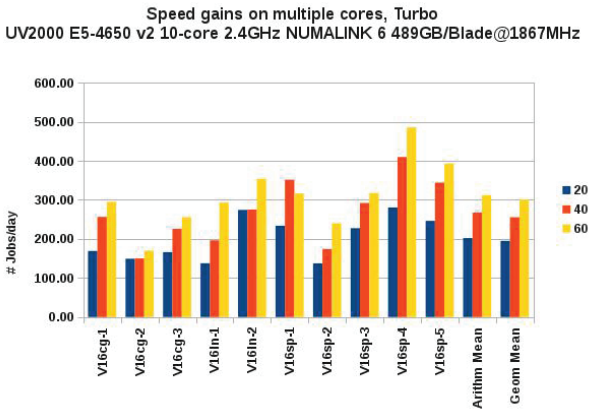

Figure 20

4.0 Conclusions

This study showed how interconnect, processor architecture, core frequency, Turbo Boost, and hyper-threading effects on performance can be gauged for ANSYS Mechanical runs. All these effects are seen to be dependent on the datasets and solution methods used. Procurement of the right mix of resources should therefore be tailored to the mix envisaged. Upgrading a single system attribute like CPU frequency, interconnect, or number of cores, diminishes returns if the others are kept unchanged. Elapsed time performance can be translated into derived metrics such as turnaround times or throughput and the cost to achieve them can be broken up into acquisition, licensing, energy, facilities and maintenance components.

5.0 References

[1] White Paper, “Obtaining Optimal Performance in ANSYS 11.” http://tinyurl.com/257n9md.

[2] SGI. SGI Developer’s Guide. Silicon Graphics International, Milpitas, California, 2009.

[3] 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.

6.0 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.

Global Sales and Support: sgi.com