WHITE PAPER

MD NASTRAN™ on Advanced SGI® Architectures*

Olivier Schreiber †, Scott Shaw †, Joe Griffin**

Abstract

This paper will explore interconnect latency and bandwidth, processor, memory and file system requirements to establish guidelines for running MD Nastran on advanced SGI computer hardware systems. The scope of the capabilities used covers Normal Mode Analysis using Shared Memory Parallelism (SMP) and Distributed Memory Parallelism (DMP) and their combination (Hybrid Mode) through geometry and frequency domain decomposition on Shared and Distributed Memory systems ranging from single node multicore workstations through multiple nodes clusters to single image many-core systems addressing very large memory space.

* Presented at MSC.SOFTWARE 2010 Automotive Conference

† SGI Applications Engineering
** Development Engineer, MSC.Software

TABLE OF CONTENTS

Introduction

How to use SGI Octane™ III, Altix®, Altix XE, Altix ICE and SGI UV embodying multiple computer technologies (Figure 1) is described in Ref [1]. They can all run MD Nastran solvers such as normal modes analysis. Software capabilities employed by MD Nastran to utilize this hardware include Shared Memory Parallel (SMP), Distributed Memory Parallel (DMP) and their combination (Hybrid Mode) as described in Ref [2]. This paper will present an overview of the two sides of these activities and how to adapt them to each other.

Figure 1: New hardware technologies

1.0 SGI hardware overview

Various systems comprised in SGI product line and available through SGI Cyclone™, HPC on-demand Cloud Computing (see section 1.7) were used to run the benchmarks described in section 3.1.

1.1 SGI Octane III

Scalable desk side multi-node system with GigE or Infiniband interconnects, up to 10 nodes, 120 cores with SUSE® Linux® Enterprise Server 10 SP2, SGI ProPack™ 6SP3.

- Dual-socket nodes of 2.93GHz six-core Xeon®X5670, 12MB cache
- Dual-socket nodes of 2.93GHz quad-core Xeon®X5570, 8MB cache
- Dual-socket nodes of 2.53GHz quad-core Xeon®E5540, 8MB cache
- RAM: 48, 72, 96GB/node 1066, 1333MHz DDR3 ECC

Figure 2: SGI Octane III

1.2 SGI Altix XE 1300 cluster

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

Figure 3: SGI Altix XE 1300 cluster

- SGI XE250 or XE270 Administrative/NFS Server node
- SGI XE340 Dual-socket compute nodes of 2.93GHz six core Xeon X5670 12MB Cache
- SGI XE340 Dual-socket compute nodes of 2.93GHz quad core Xeon X5570 8MB Cache
- SGI XE250 Dual-socket compute nodes of 3.0GHz quad core Xeon X5472 12MB Cache, 1600MHz Front Size Bus.
- 32GB 1333MHz RAM(max 96GB/node)
- SUSE Linux Enterprise Server 11 SP2, SGI ProPack 6SP3
- SGI Foundation Software 1SP5
- Infiniband ConnectX QDR PCIe Host Card Adapters
- Integrated GigE dual port Network Interface Cards

SGI Altix XE 1300 cluster with dual Ethernet and Infiniband switch is illustrated in Figure 4.

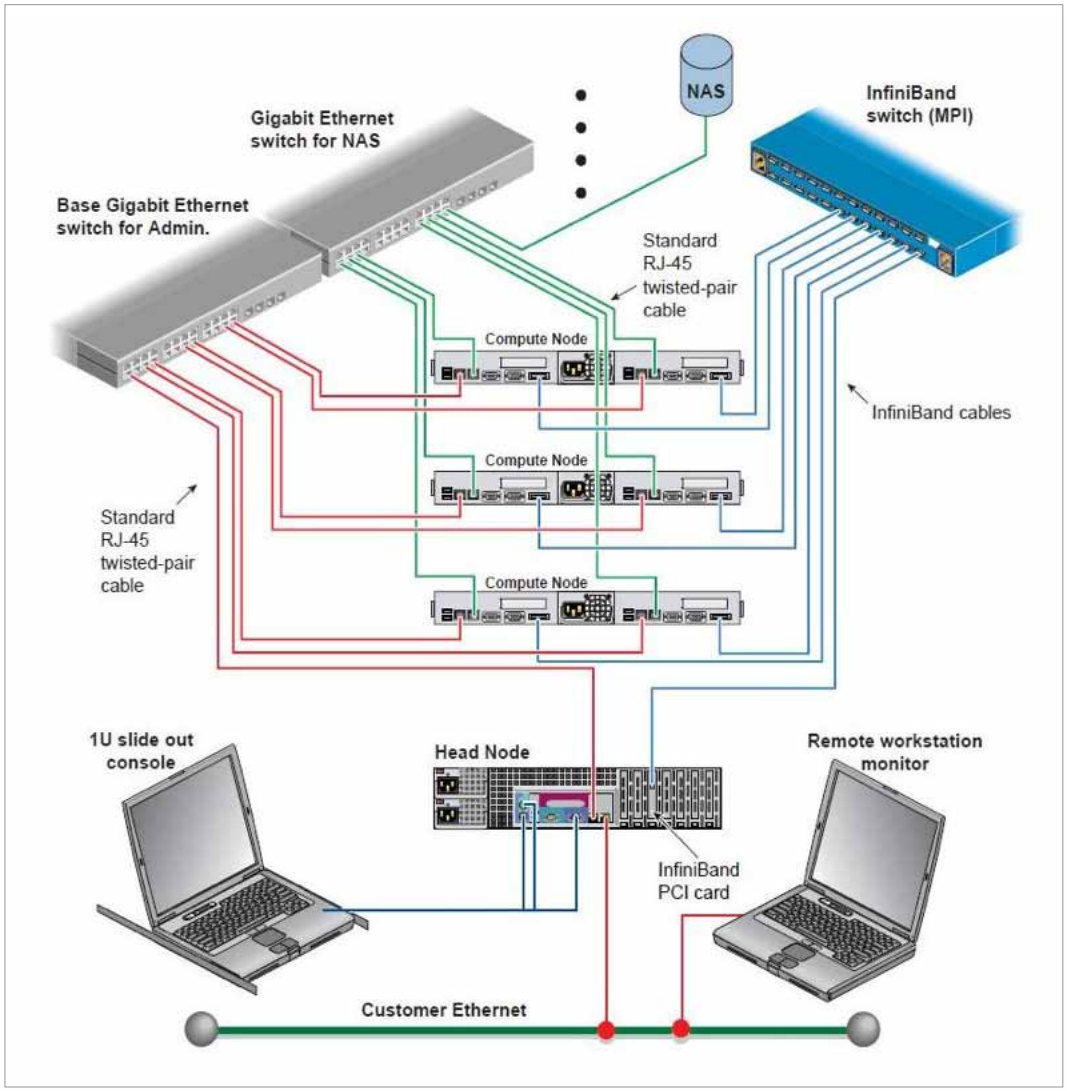

Figure 4: Dual Ethernet and Infiniband switch cluster configuration example

1.3 SGI Altix ICE cluster

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

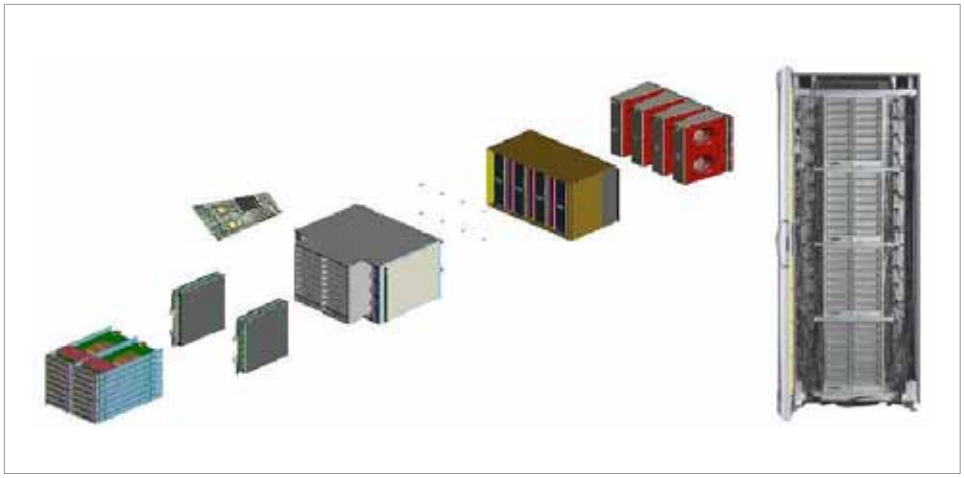

Figure 5: SGI Altix ICE cluster and IRU

- Intel Xeon 5500 2.93GHz quad-core or 5600 3.46GHz six-core
- Two single-port ConnectX-2 IB HCA
- 12 DDR3 1066 MHz or 1333 MHz ECC DIMM slots per blade
- SGI Tempo management tools
- SGI ProPack™ for Linux®
- Altair® PBS Professional™ workload manager

1.4 SGI Altix 450 and Altix 4700 (SMP)

A highly scalable Shared Memory Parallel (SMP) system allows flexibility of sizing memory allocated to a job independently from the core allocation. In a multi-user, heterogeneous workload environment, this prevents jobs requiring a large amount of memory to be 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 there is at worst the option to run the job serially on 1 core with 128GB RAM allocation.

Figure 6: SGI Altix 4700, 450 SMP

- SGI Altix 4700 1.669GHz dual core Itanium® 9150M 24MB Cache processors, SGI NUMAlink® 4
- SUSE Linux Enterprise Server 11 SP2, SGI® ProPack 6SP3 for Linux®

1.5 SGI Altix UV 10, UV 100, UV 1000 SMP

Highly scalable latest generation x86-based Shared Memory Parallel system. Affords the same flexibility as the architecture of 1.4.

Figure 7: SGI Altix UV 10, UV 100, UV 1000 SMP

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

1.6 Altix XE, Altix ICE are Intel Cluster Ready Certified

Altix XE, Altix ICE are Intel® Cluster Ready Certified (Figure 8).

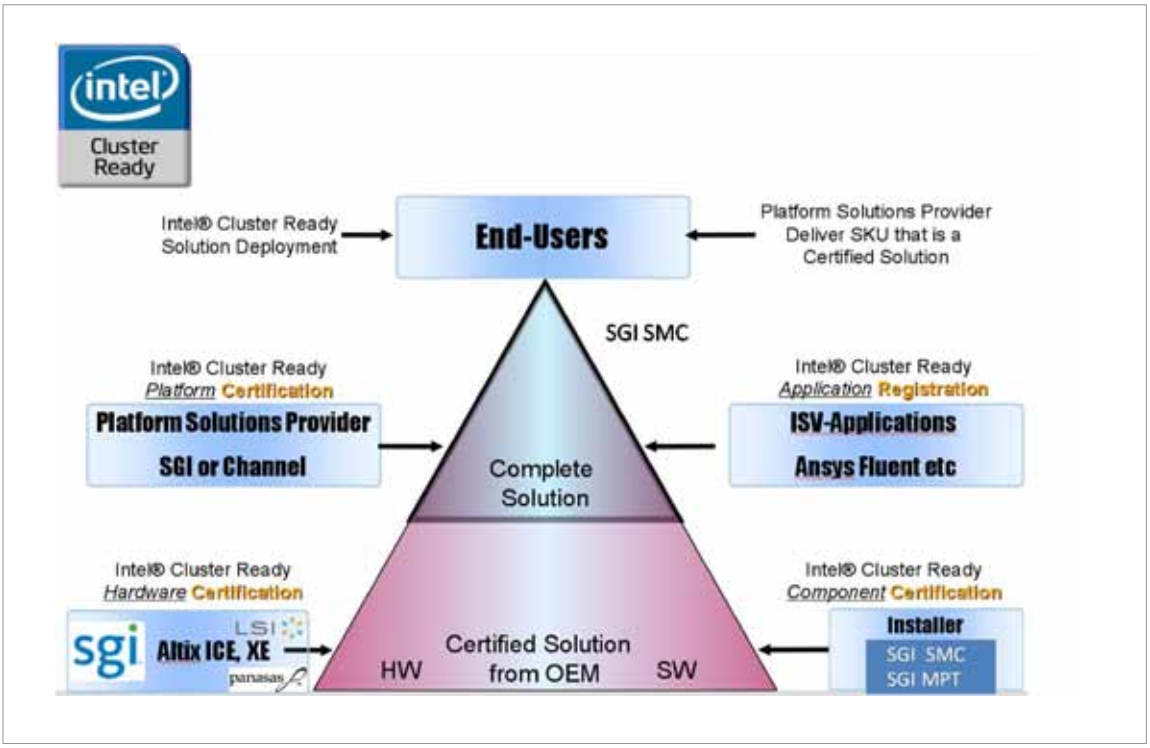

Figure 8: Intel Cluster Ready Certification

1.7 Cloud access to benchmark systems: SGI Cyclone™

SGI offers Cyclone, HPC on-demand computing resources of all SGI advanced architectures aforementioned (Figure 9). There are two service models in Cyclone: Software as a Service (SaaS) and Infrastructure as a Service (IaaS) (Figure 10). With SaaS, Cyclone customers can significantly reduce time to results by accessing leading-edge open source applications and best-of-breed commercial software platforms from top Independent Software Vendors (ISV's).

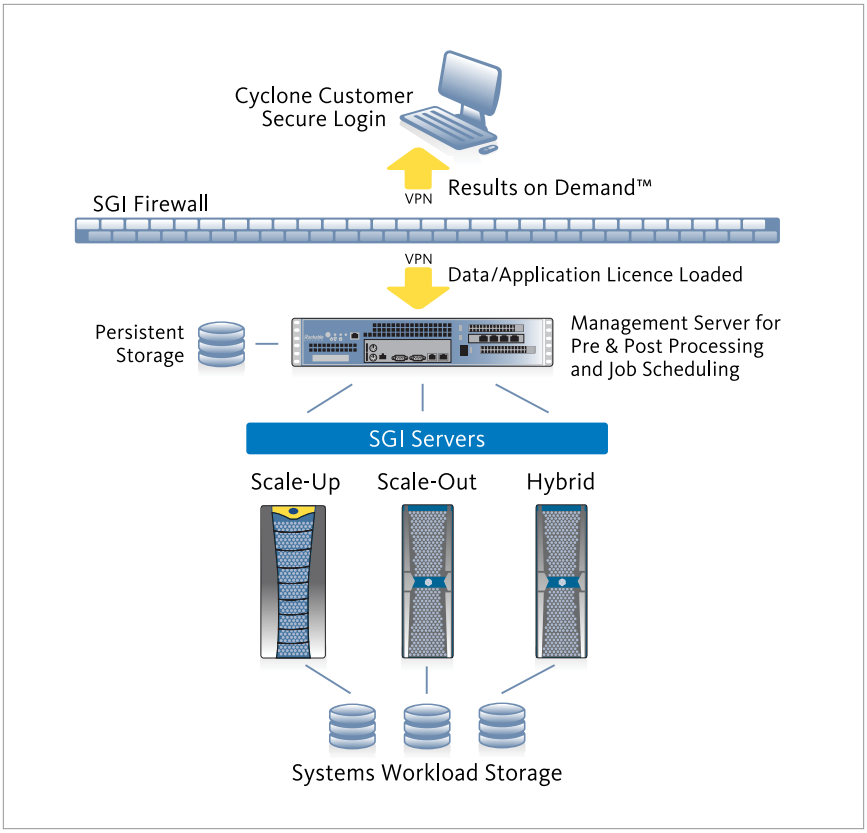

Figure 9: SGI Cyclone – HPC on-demand Cloud Computing

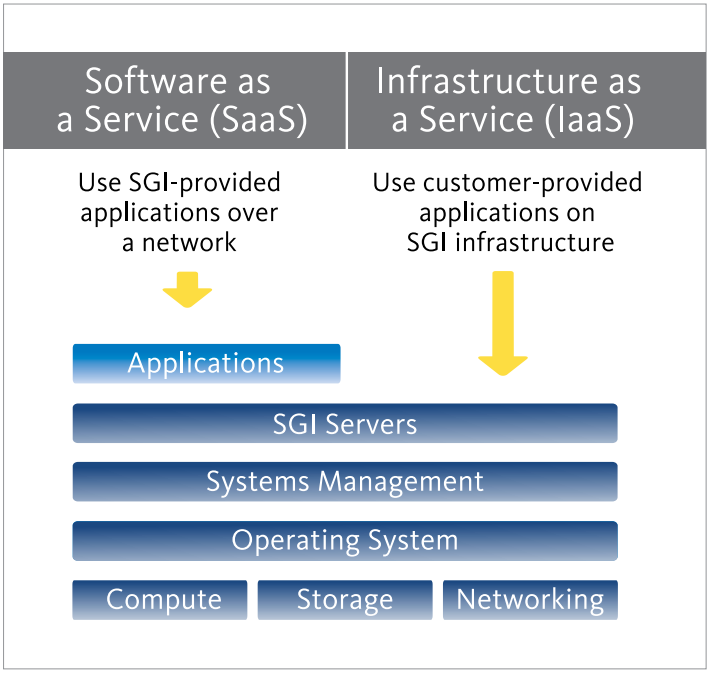

Figure 10: SGI Cyclone – Service Models

The physical elements of Cyclone features:

- Pre-configured, pre-certified applications and tools
- High speeds Scale-Up, Scale-Out and Hybrid(GPU/Graphics) platforms
- High speed processors
- High speed networking (NUMAlink, InfiniBand)
- Non-virtualized environments
- Dedicated management node for security
- SSH or Virtual Private Network(VPN) access
- From scratch storage to long-term storage
- Data exchange service
- 24 x 7 x 365 monitoring and support
- Dedicated customer accounts
- SGI professional services also available

2.0 MD Nastran Usage for the study

2.1 Parallelism

To avoid confusion with the concepts involving various computer systems, a specific nomenclature is used here for clarity. A node or host means a computer system associated with one network interface and its address, implemented on a board enclosed in a rack-mounted chassis or blade. This board comprises two sockets (most common) or more receiving the processor with the 4, 6, 8 or 12 cores constituting the actual Central Processing Units (CPU's). Shared Memory Parallelism (SMP) appeared in the 1980's for 'DO Loop' processing strip mining or subroutine spawning via memory-sharing threads. In the late 1990's Domain Decomposition Parallel (DMP) Processing appeared and revealed itself more suitable for performance gains because of coarser grain parallelism. In the mean time, Shared Memory Parallelism was maintained as is but saw the adjunction of mathematical libraries already parallelized using efficient implementation of Shared Memory Parallelism API OpenMP™ (Open Multi-Processing). Both parallel computing methods can run on Shared Memory systems like a single host or distributed memory systems like clusters but the constraints are different: Shared Memory Processing cannot span cluster nodes both communication and memory-wise. The two methods can be combined together in what is called 'Hybrid Mode'. However, performance gains may not always be realized without attention to the many ways to combine these methods.

2.2 Shared Memory Parallel Processing

Shared Memory Parallelism (SMP)–meaning memory shared by all cores of each node– uses OpenMP Application Programming Interface™ (Open Multi-Processing) or Pthreads and is simply activated by invoking smp=<value> on the command line.

2.3 Distributed Memory Parallel Processing

These methods are activated by invoking dmp=<value> on the command line, and as many MPI processes are distributed in 'rank' or round-robin allocation across all hosts=<host:host:...> designated nodes on the multiple cores.

2.3.1 Geometry Domain Decomposition

In this method activated by simply invoking dmp=<value> on the command line, the finite element domain or global matrices is partitioned into as many MPI processes.

2.3.2 Frequency Domain Decomposition

This method, activated by invoking numseg=<dmparallel_value> on the command line, partitions the frequency domain into as many MPI processes.

2.4 Execution

Submittal procedure must ensure:

- Placement of processes and threads across nodes and sockets within nodes
- Control of process memory allocation to stay within node capacity
- 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. To achieve the best runtime in a batch environment disk access to input and output files should be placed on the high performance file system closest to the compute node. The high performance file system could be in-memory file system (/dev/shm), a Direct (DAS) or Network (NAS) Attached Storage file system. In diskless computing environments in-memory file system or network attached storage are the only options. This file system nomenclature is illustrated in Figure 11.

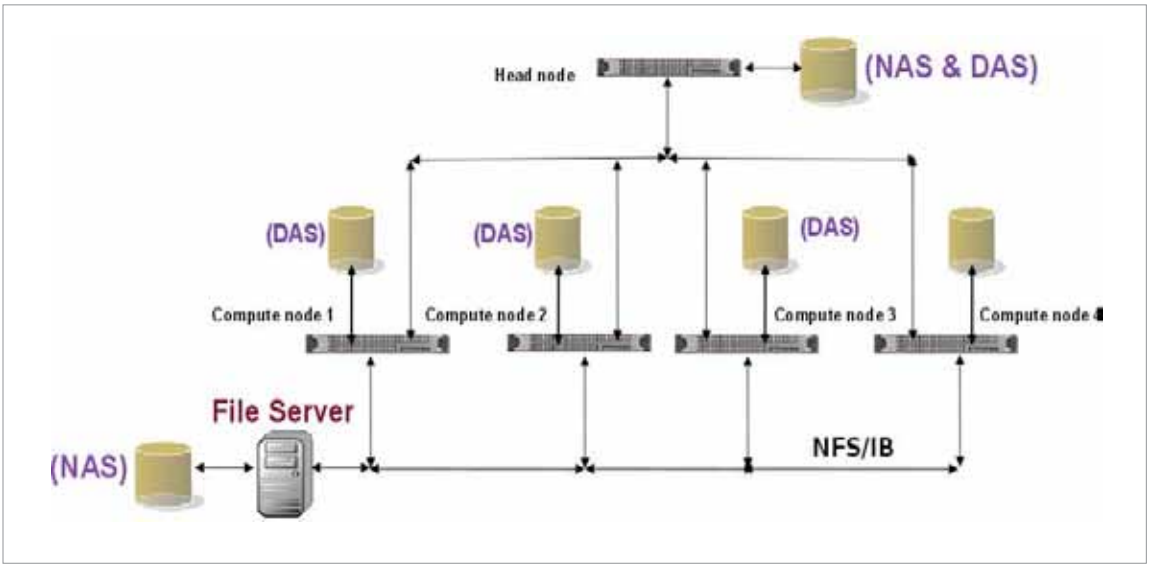

Figure 11: Example file systems for scratch space

Following is the synoptic of a job submission script.

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

2.5 Software Environment

MSC.Nastran 2008 and MD Nastran 2010.1 were used.

3.0 Results

3.1 Benchmark example

The benchmark used is a Car Body model similar to Figure 12. It has approximately 268,000 Grids, 275,000 Elements, 1,584,000 degrees of freedom. The solution sequence performed is solution 103 with all roots requested in the frequency range of 0 to 200Hz and 0 to 400Hz, that is approximately 1000 and 3000 modes, respectively.

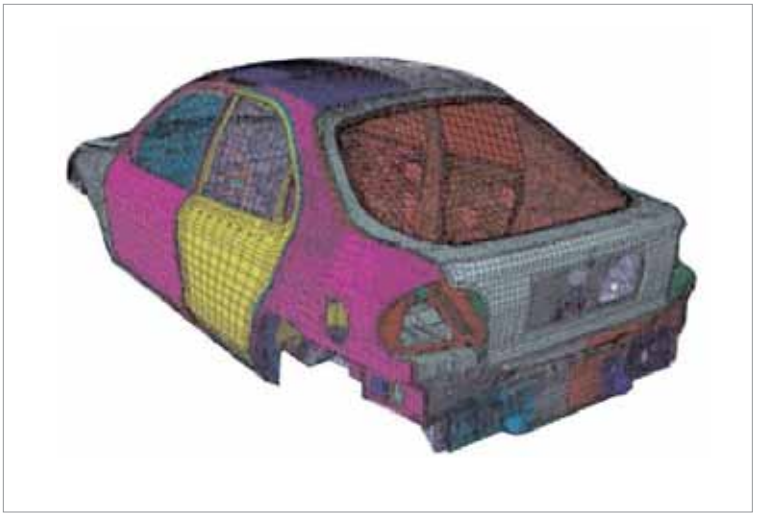

Figure 12: Car Body Model

3.2 Benchmark Results

3.2.1 Scaling on 1 node

Figure 13 shows scaling within one 12-core X5670/2.93GHz node in configurations similar to 1.1 or 1.2 for this standard benchmark run in DMP (MPI) mode parallelism using two different domain decompositions, namely Geometry Domain Decomposition (2.2.3) and Frequency Domain Decomposition (2.2.1). One can see that the optimal number of processes is different for the two methods and an anomaly exists for a 6-way Geometry Domain Decomposition. Beyond 8-way decomposition, elapsed times increase and therefore Hyperthreading, i.e. using more than the number (12) of physical cores available would not be beneficial. (Hyper-Threading (HT) is a feature which can increase performance for multi-threaded or multi-process applications. It allows a user to run twice the number of OpenMP threads and Pthreads or MPI processes than available physical cores per node.) The domain decomposition used does not necessarily cause a proportional division of the memory and I/O bandwidth requirements on the individual cores processing each domain. Indeed, Geometry Domain Decomposition (2.2.3) and Frequency Domain Decomposition (2.2.1) are different in this regard and the solution sequence used also affects the resource requirements changes. Moreover, the user can modify with a posteriori knowledge of the changed resource requirements the explicit memory allocation and/or usage of MSC Nastran I/O buffering and therefore change the performance of execution and optimal number of processes.

The trend, though is that beyond a certain number of processes, the aggregate memory and memory bandwidth demand will grow beyond that which is available from the node. In addition, communication needs increase between the different domains and further slow down the potential speedup.

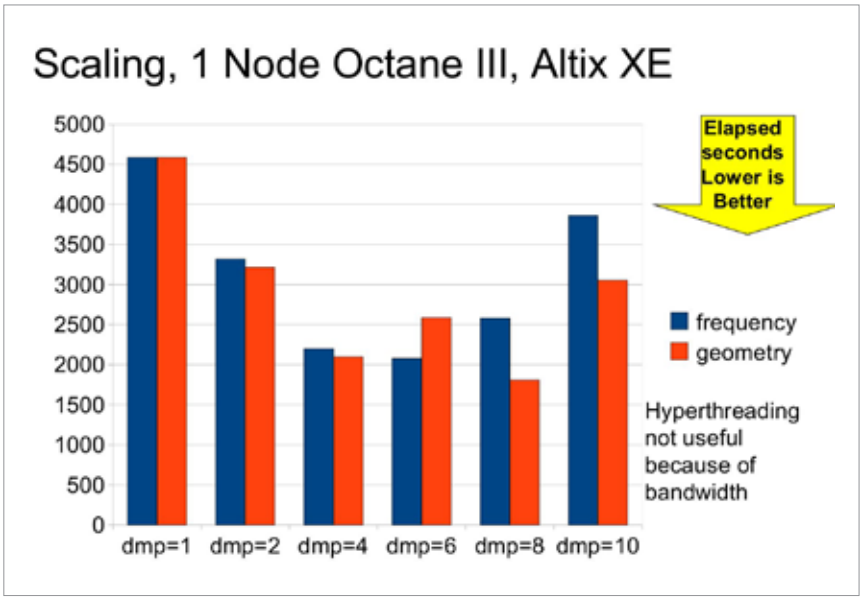

Figure 13: Scaling 1 node

3.2.2 Scaling on multiple nodes

To overcome the limitation shown in 3.2.1, one can resort to using multiple nodes. Choosing the Geometry Domain Decomposition method (2.2.3) as an example, figure 14 shows how scaling can be increased beyond 8 processes with a corresponding reduction of elapsed time on 12-core X5670/2.93GHz nodes in configurations similar to 1.1 or 1.2. Beyond 16 processes for this data set and decomposition method, one finds that scaling will not continue, regardless of the number of nodes used as can be seen by increased elapsed times for 32 processes.

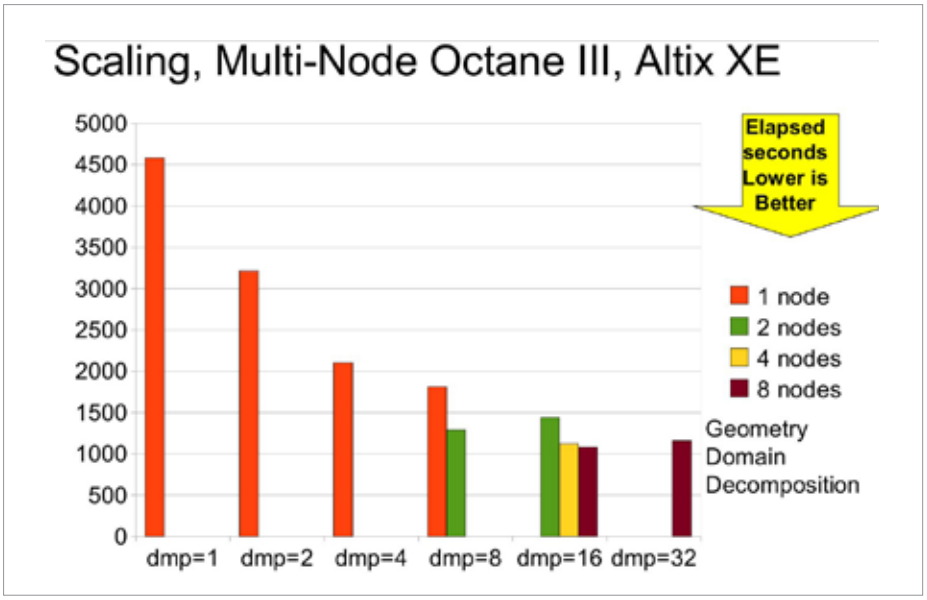

Figure 14: Scaling on multiple nodes

3.2.3 Influence of Communication Protocol and Interconnect

When using more than 1 node for a Distributed Memory Parallel (MPI) computation, the choice of Interconnect is available between GigE and Infiniband. It is also possible to run TCP/IP protocol on Infiniband. Figure 15 shows that performance difference can be dramatic for 4 nodes in configurations similar to 1.1 or 1.2.

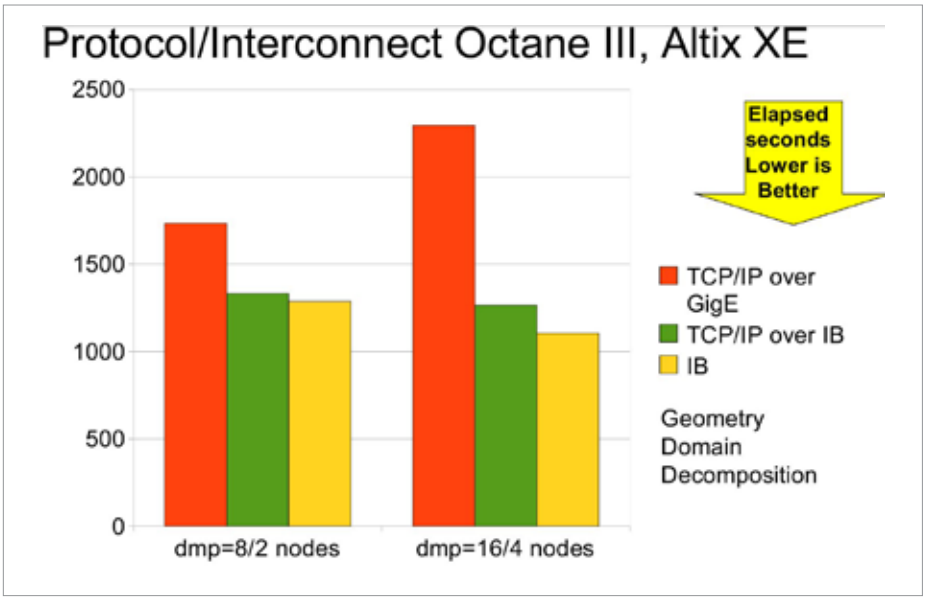

Figure 15: Influence of Communication Protocol and Interconnect

3.2.4 SGI Message Passing Toolkit (MPT) MPI library through PerfBoost

An MPI library capability to bind an MPI rank to a processor core is key to control performance because of the multiple node/socket/core environments. From [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, MPT, through the omplace command uniquely provides convenient placement of hybrid MPI/OpenMP processes and threads and Pthreads within each node. This MPI library is linklessly available through the PerfBoost facility bundled with SGI ProPack. PerfBoost provides a Platform-MPI, IntelMPI, OpenMPI, HP-MPI ABI-compatible interface to MPT MPI. Figure 16 shows that without the comparative advantage in handling hybrid mode, MPT is on par or better than competing MPI libraries in various MPI runs in Geometry Domain Decomposition method (2.2.3) in configurations similar to 1.1 or 1.2.

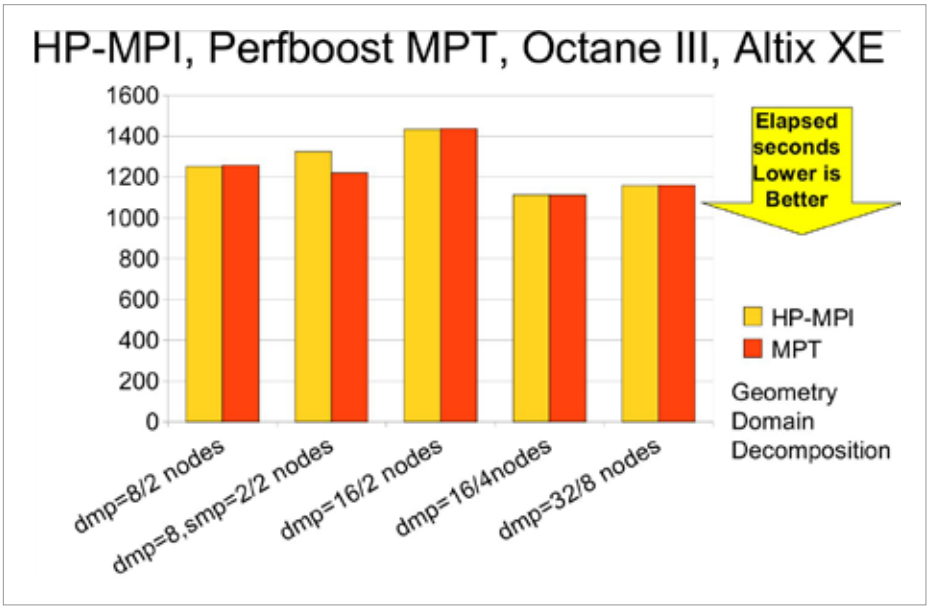

Figure 16: SGI Message Passing Toolkit (MPT) MPI library through PerfBoost

3.2.5 Effect of core frequency and Intel Turbo Boost Technology

Figure 17 plots elapsed times for increasing core frequencies starting from a base of 2.53GHz up to 2.93GHz for the serial run case and 8-way parallel Geometry Domain Decomposition method (2.2.3) in configurations similar to 1.1 or 1.2. A reference plot is drawn for what the elapsed time should have been, ideally, had it been inversely proportional to the frequency. It is apparent that MD Nastran in these two cases at least is not core processing speed bound and therefore, a trade off may exist between faster cores and how many can be procured, considering their cost.

An additional data point regards the added performance gained through Turbo Boost. Intel Turbo Boost is a feature first introduced in the Intel Xeon 5500 series, for increasing performance by raising the core operating frequency within controlled limits depending on thermal envelope. The mode of activation is a function of how many cores are active at a given moment as maybe the case when OpenMP threads or Pthreads or MPI processes are idle under their running parent. For example, for a base frequency of 2.93GHz, when 1-2 cores active, their running frequencies will be throttled up to 3.3GHZ, but with 3-4 cores active only to 3.2GHz. For most computations, utilizing Turbo Boost technology can result in improved runtimes, but the overall benefit may be mitigated by the presence of other performance bottlenecks than pure arithmetic processing. The performance gain is predictably higher for the serial run (3%) than for the 8-way parallel run (.5%).

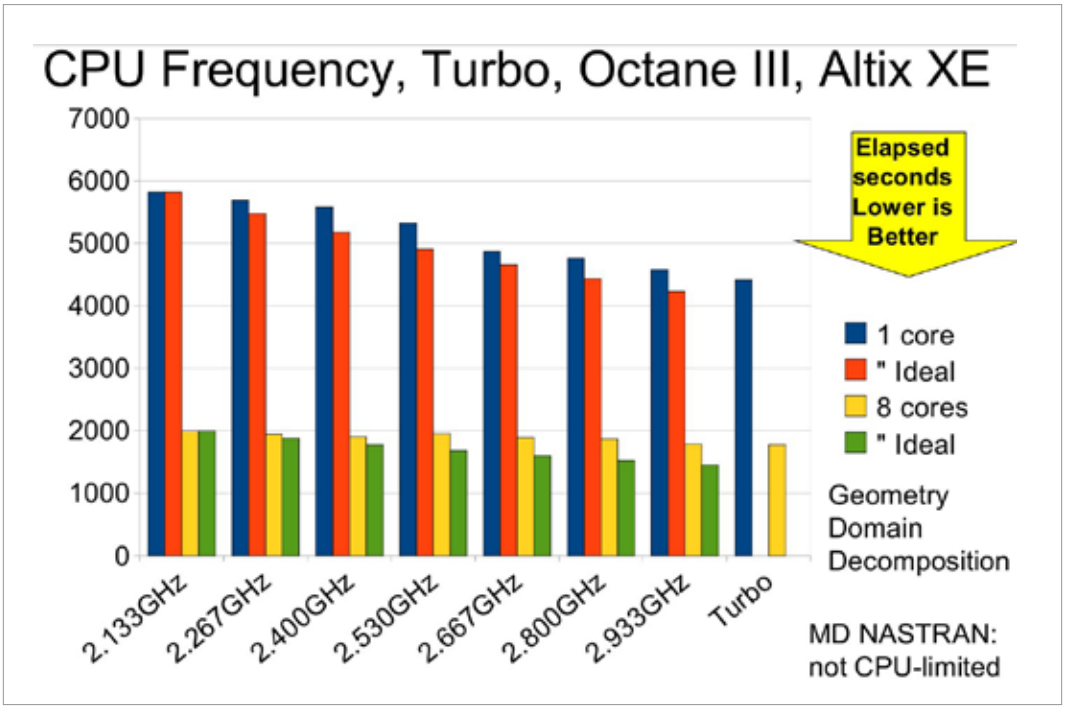

Figure 17: Effect of core frequency and Turbo Boost

3.2.6 Effect of the scratch space file system

An I/O dominated application relies on a high performance file system for best performance. The I/O subsystem either being DAS or NAS needs to be configured to support fast I/O sequential transactions as illustrated in section 2.3.1 by Figure 11. In cluster computing environments with a common scratch location, such as a Network Attached File system (NAS), isolating application MPI communications and NFS traffic will provide the best NFS I/O throughput for scratch files.

Figure 18 shows changing scratch space file system location may have dramatic effect on performance in any run case situation as in Geometry Domain Decomposition method (2.2.3) in configurations similar to 1.1 or 1.2. The case of 1 node run did not allow scratch space to fit into RAM.

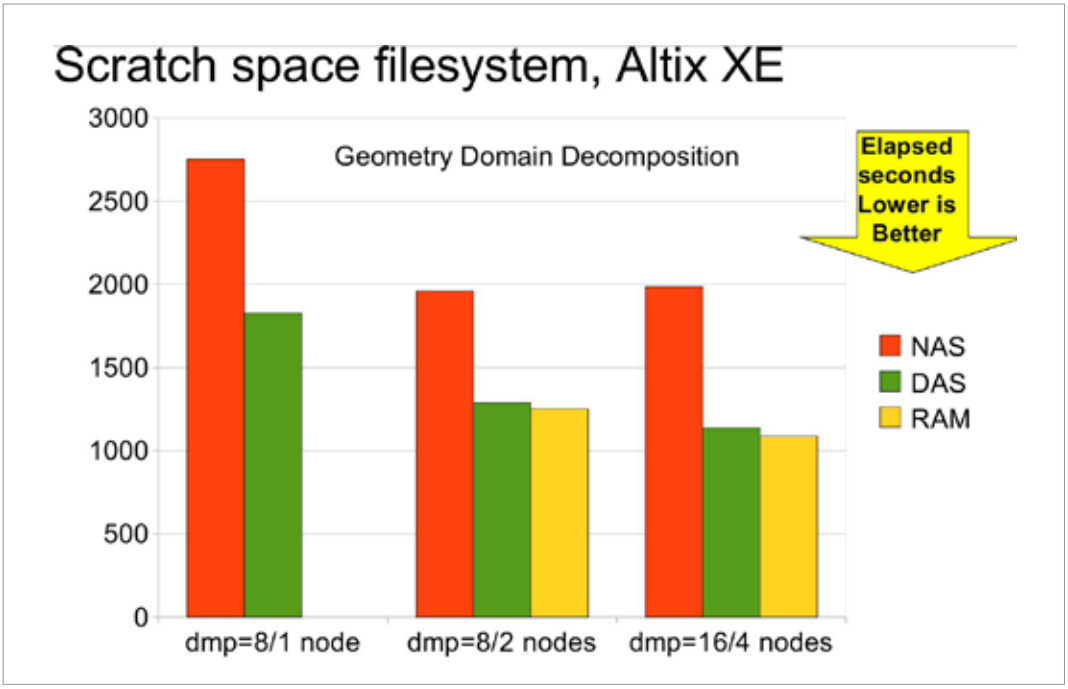

Figure 18: Effect of scratch space file system

3.2.7 Effect of libFFIO (Flexible File I/O)

SGI's FFIO is a linkless library (which does not need to be linked to the application) bundled with ProPack which implements user defined I/O buffer cache to avoid memory buffer cache thrashing when running multiple I/O intensive jobs or processes 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 and 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 [2], Chapter 7 Flexible File I/O). Figures 19 and 20 show that whereas libFFIO may not provide a benefit for Geometry Domain Decomposition method (2.2.3) runs on a Direct Attached file system (DAS) on configurations similar to 1.1 or 1.2, it does improve performance on a Network Attached file system (NAS). The strategy to employ is to use the static mem= memory allocation for Nastran to run in-core and then allocate what is left over for libFFIO. This needs to be done per process.

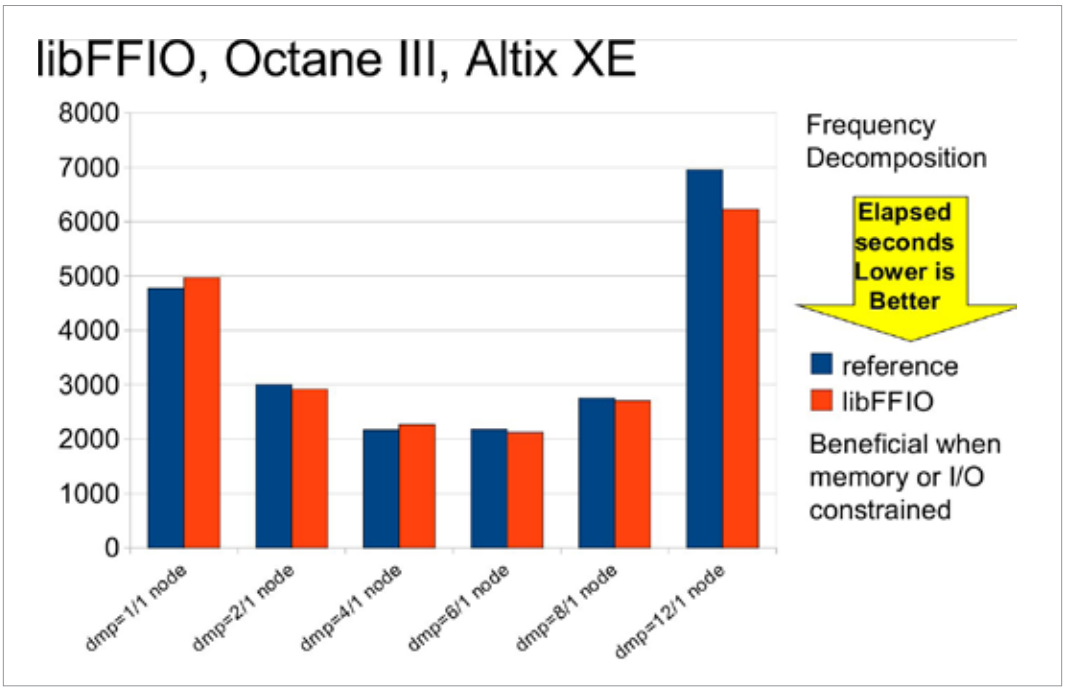

Figure 19: Effect of LibFFIO

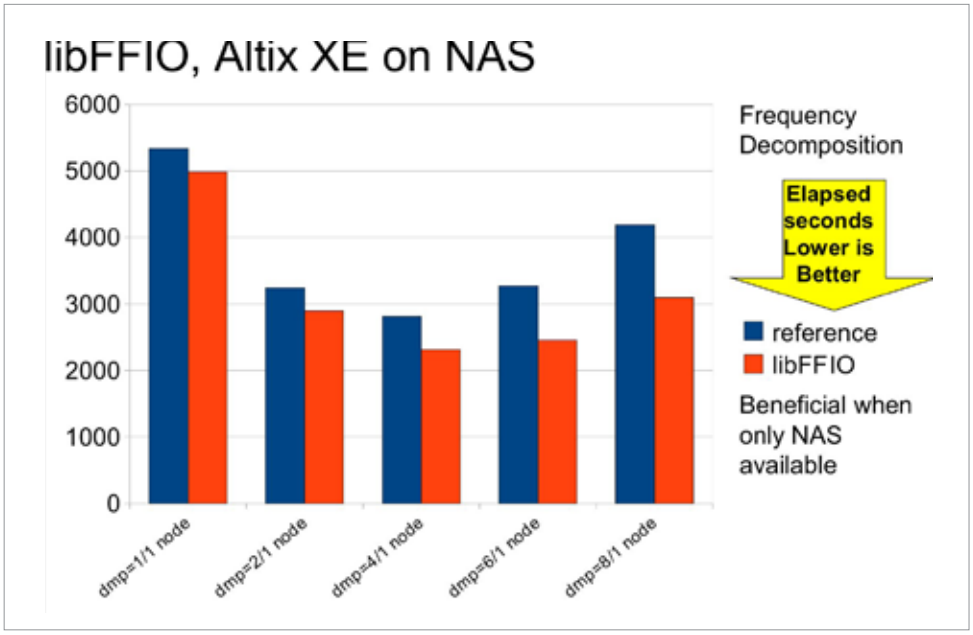

Figure 20: Effect of libFFIO with Network Attached File system (NAS)

3.2.8 Effect of RAM available on node

For a Distributed Memory Parallel (DMP) job (MPI), using a given Decomposition Method on 1 node, the aggregate memory and disk usage requirements will be approximately constant with varying number of processes used. However, Figures 21 and 22 show that because Geometry Domain Decomposition (2.2.3) has a high water mark for scratch space usage almost fitting within 48GB Linux buffer cache but Frequency Domain Decomposition (2.2.1) has a high water mark for scratch space usage which is twice the former, additional RAM in the second case will allow Linux Buffer cache to accommodate more I/O, resulting in increased performance. (The configuration is similar to 1.1 or 1.2).

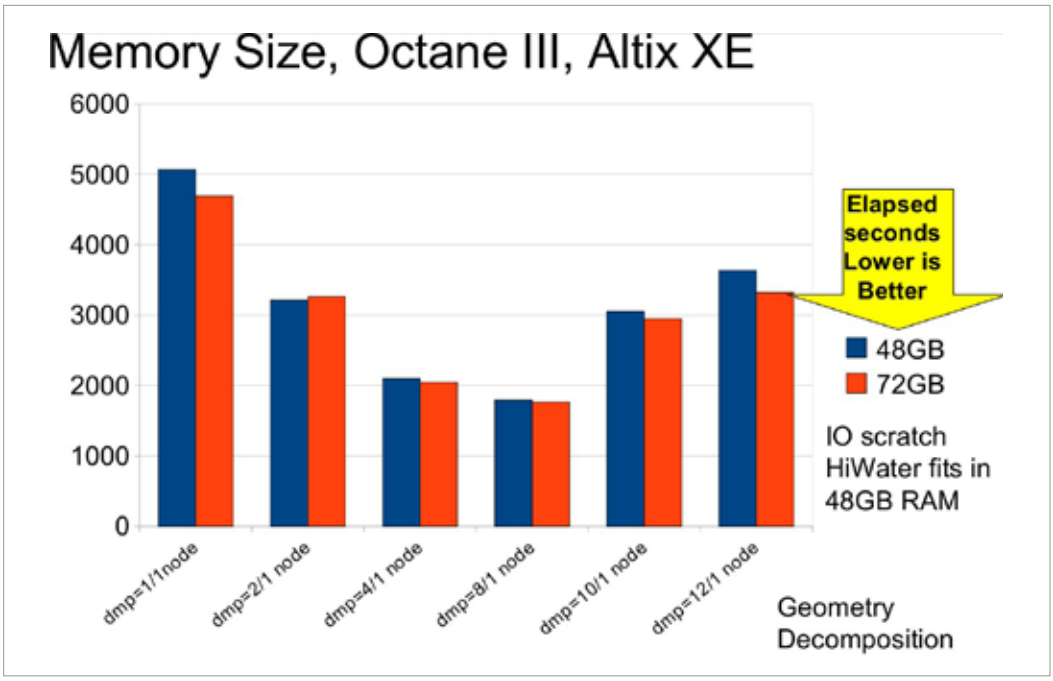

Figure 21: Effect of RAM available on node for Geometry Decomposition Method

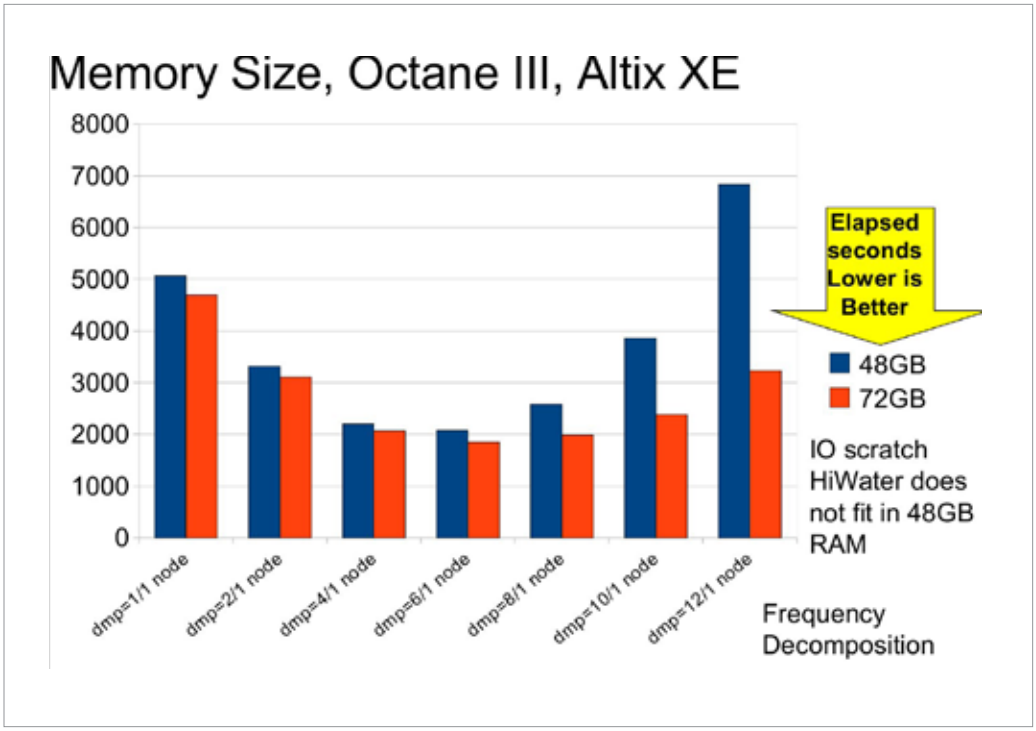

Figure 22: Effect of RAM available on node for Frequency Decomposition Method

3.2.9 Effect of memory speed

To isolate the effect of memory speed, the ACMS method was run with 2 processes on 1 node so that RAM could be used to accommodate the smaller scratch space file system requirements. No discernable performance was found between 800MHz and 1333MHz rated memory. The explanation is that core to RAM bandwidth is the limiting factor in this case as opposed to RAM latency.

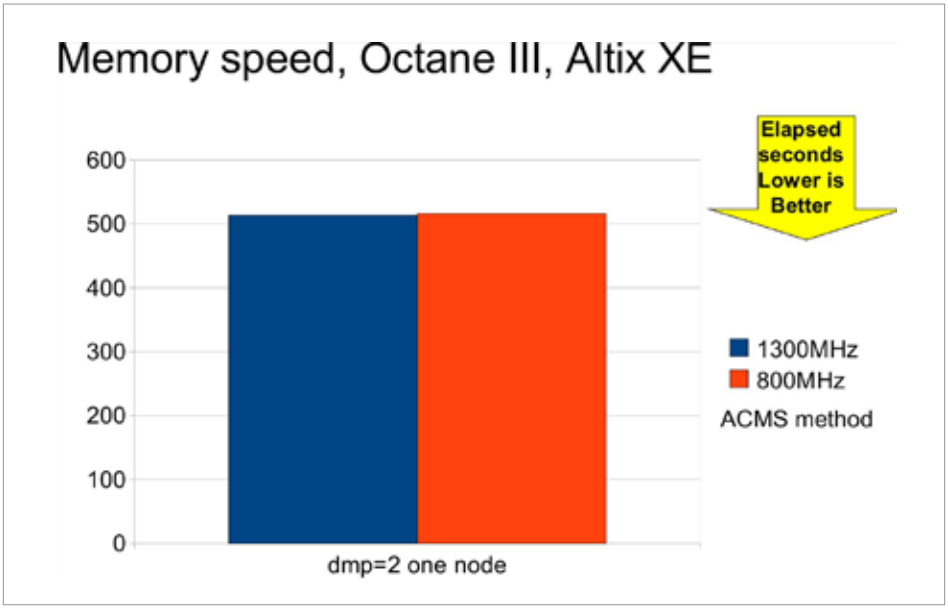

Figure 23: Effect of memory speed

3.2.10 Combining SMP with DMP or hybrid mode using PerfBoost and MPT

Shared Memory Parallelism mode is implemented with threads through OpenMP API or directly with Pthreads, outside I/O processing regions so parallel efficiency is limited because of the fine parallel granularity but I/O requirements do not rise. Therefore combining advantages of both paradigms by running SMP simultaneously with DMP using SGI's MPT and omplace as outlined in 3.2.4 yields valuable performance gains as shown in Figure 24 on configurations similar to 1.5.

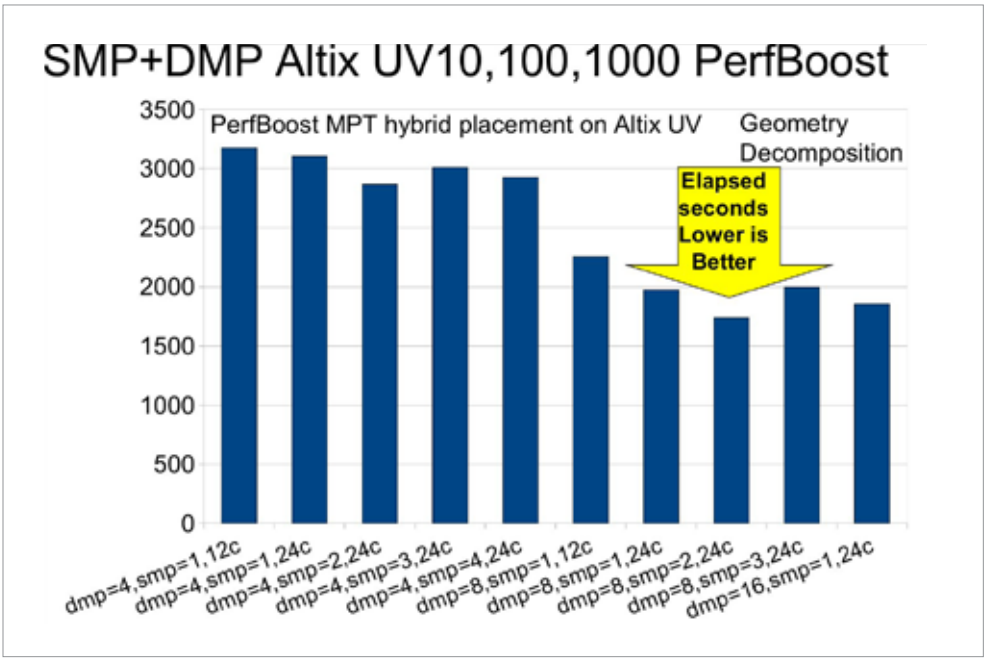

Figure 24: Combining SMP with DMP or hybrid mode using PerfBoost and MPT

Conclusions

Interconnect and associated libraries, core frequency, Turbo Boost, Hyper-Threading, file system, memory and memory speed, parallelism effects on performance can be gauged for a given dataset. In particular one observed:

- Infiniband can be twice as fast as GigE interconnect.
- MPT (through PerfBoost) compares favorably to other MPI's in all cases.
- Effect of Frequency and Turbo Boost are weak as this is not the only limiting factor for performance.
- HyperThreading does not help because of communication costs beyond 8 processes.
- DAS or even RAM is always faster than NAS.
- libFFIO to be used only when I/O and RAM constrained as in small DAS or NAS.
- More RAM accommodates avails more Linux buffer cache
- DIMM speed effect is negligible in the particular case which was studied.
- Good scaling for SMP and DMP beyond one node/16 cores.

All these effects are definitely dependent on the dataset and solution methods used. Procurement of the right mix of resources should therefore be tailored to the range of datasets envisaged. Moreover, the metric to minimize could be one of many such as turnaround time or throughput or cost–itself comprised of acquisition cost, licenses, energy, facilities and services.

Attributions

MD Nastran is a registered trademark of MSC.Software Corporation. SGI, Octane, Altix, ProPack and Cyclone are registered trademarks or trademarks of Silicon Graphics International Corp. or its subsidiaries in the United States or other countries. Xeon and Itanium are trademarks or registered trademarks 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.

References

[1] C. Liao, O. Schreiber, S. Kodiyalam, and L. Lewis. “Solving Large Problems on Large Computer Systems. Hardware Considerations for Engineering Productivity Gains with Multi-Discipline MSC NASTRAN Simulations”. Presented at MSC.Software® VPD Conference, October 11-12 2007.

[2] SGI Linux Application Tuning Guide, Silicon Graphics International, Fremont, 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.

Global Sales and Support: sgi.com/global

©2011-2012 Silicon Graphics International Corp. All rights reserved. SGI and the SGI logo are registered trademarks or trademarks of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries. 10012012 4262