TECH GUIDE

MSC NASTRAN EXPLICIT NONLINEAR (SOL 700) ON ADVANCED SGI® ARCHITECTURES

Authors

Dr. Olivier Schreiber (SGI), Walter Schrauwen Msc Software,
Tony Devarco (SGI), Scott Shaw (SGI), Daniel Thomas (SGI)

Summary

MSC Nastran Explicit Nonlinear (SOL 700) provides engineers a powerful structural analysis solution to model high velocity or short duration events such as impact, crash, explosions, and numerous others. During an explicit nonlinear analysis, at each time step, the MSC Nastran program couples LSTC LS-DYNA FEA structure/contact locations and velocities with forces from MSC Software's Dytran Fluid Structure Interaction (FSI) simulation. Each separate solver capability is still available separately if needed and parallel processing of both supports all three use cases. This paper will explore interconnect latency and bandwidth, processor, memory and filesystem requirements to establish guidelines for running MSC Nastran Explicit Nonlinear (SOL 700) on advanced SGI computer hardware systems using Distributed Memory Parallelism (DMP) on Shared and Distributed Memory systems ranging from single multicore nodes through multiple nodes clusters to single image many-core systems addressing very large memory space.

TABLE OF CONTENTS

1.0 Hardware Systems

Using SGI ICE, SGI Rackable, and SGI UV and associated technologies (Figure 1) is described in (SGI, 2009). All can run MSC Nastran Explicit Nonlinear (SOL 700) with its Shared Memory Parallel (SMP), Distributed Memory Parallel (DMP) and combination (Hybrid Mode) technologies (Liao, et al., 2007).

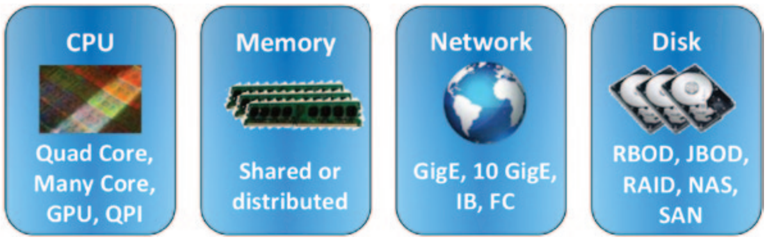

Figure 1: New hardware technologies

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

1.1 SGI® Rackable Cluster

SGI Rackable cluster supports up to 256GB of memory per node in a dense architecture with up to 32 cores per 1U with support for Linux®, FDR and QDR Infiniband, eight-core processors, GPU's and DDR3 memory (Fig.2). Configuration used for the benchmarks was:

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

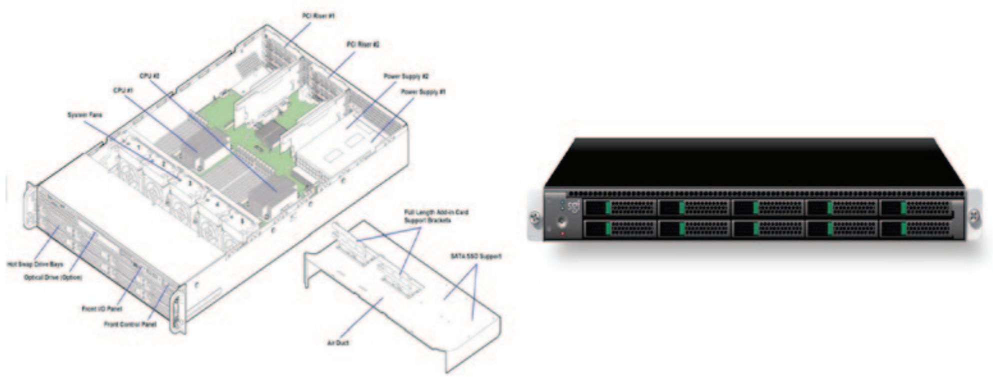

Figure 2: Overhead View of SGI Rackable Server with the Top Cover Removed and Actual Server

1.2 SGI® ICE™ X

SGI® ICE X integrated blade cluster is a highly scalable, diskless, cable-free infiniband interconnect high density rack mounted multi-node system. ICE X combines Intel® Xeon® processor E5-2600 series platform with a unique board and interconnect design. Running on standard Linux®, SGI ICE X delivers over 53 teraflops per rack of 2,304 processor cores (Fig. 3). Configuration used for the benchmarks was:

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

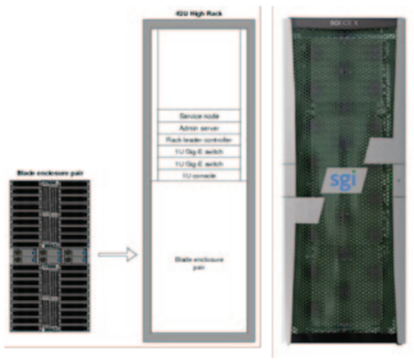

Figure 3: SGI® ICE X Cluster with Blade Enclosure

1.3 SGI® UV™ 2000

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

CAE Workflow can be accelerated for overall time to solution by running pre/Post-processing, solvers and visualization on one machine without moving data (Figure 4). Flexibility of sizing memory allocated to a job independently from the core allocation in a multi-user, heterogeneous workload environment prevents jobs requiring a large amount of memory from being starved for cores. For example, a job requiring 128GB to run in-core could be broken up through domain decomposition into 8 parallel MPI processes needing only 16GB so one could run it on 8 24GB cluster nodes. But these 8 cluster nodes may not be available in a busy environment so the job would be waiting in the queue, effectively starved for nodes. On the Shared Memory Parallel system, one can always find 8 free cores and allocate the 128GB to them for the job and there is also the option to run the job serially on 1 core with 128GB allocation.

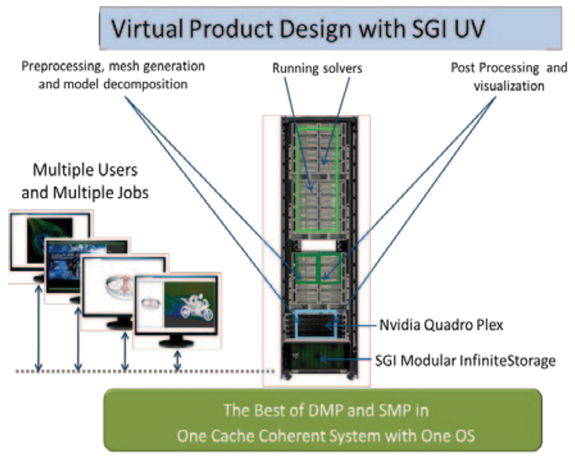
Virtual Product Design with SGI UV
Preprocessing, mesh generation and model decomposition
Running solvers
Post Processing and visualization
Multiple Users and Multiple Jobs
Nvidia Quadro Plex
SGI Modular InfiniteStorage
The Best of DMP and SMP in One Cache Coherent System with One OS

Figure 4: SGI® UV CAE Workflow

Configuration used for the benchmarks was:

- 64 sockets (512 cores) per rack
- Intel® Xeon® 8 core 2.7GHz E5-4650
- SGI NUMALink™ 6 Interconnect
- 4 GB of Memory/core
- Altair® PBSPro Batch Scheduler with CPUSET MOM v11
- SLES or RHEL with latest SGI Performance Suite, Accelerate

1.4 Access to benchmark systems

SGI offers Cyclone, HPC on-demand computing resources of all SGI advanced architectures aforementioned (Figure 5.1). Cyclone customers can reduce time to results by accessing open source applications and commercial software from Independent Software Vendors (ISV's).

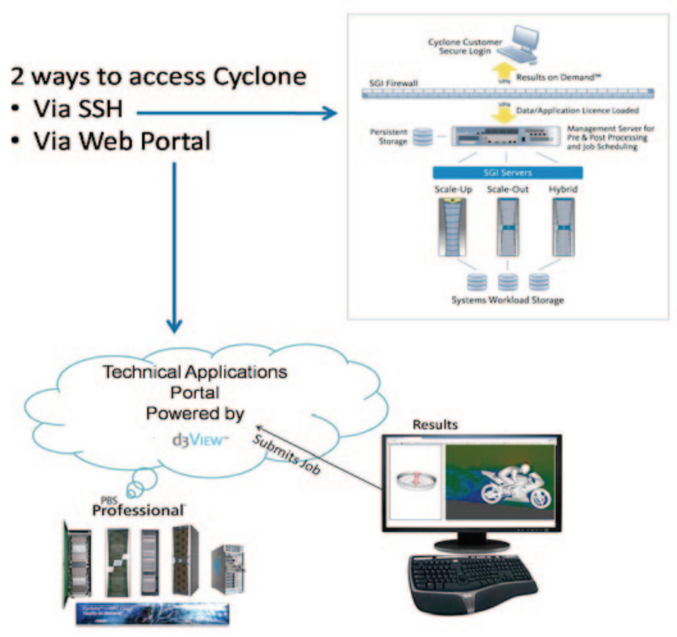

Figure 5.1: SGI Cyclone – HPC on-demand Cloud Computing

The solution to CAE workflow brought by Cyclone is illustrated by Figure 5.2

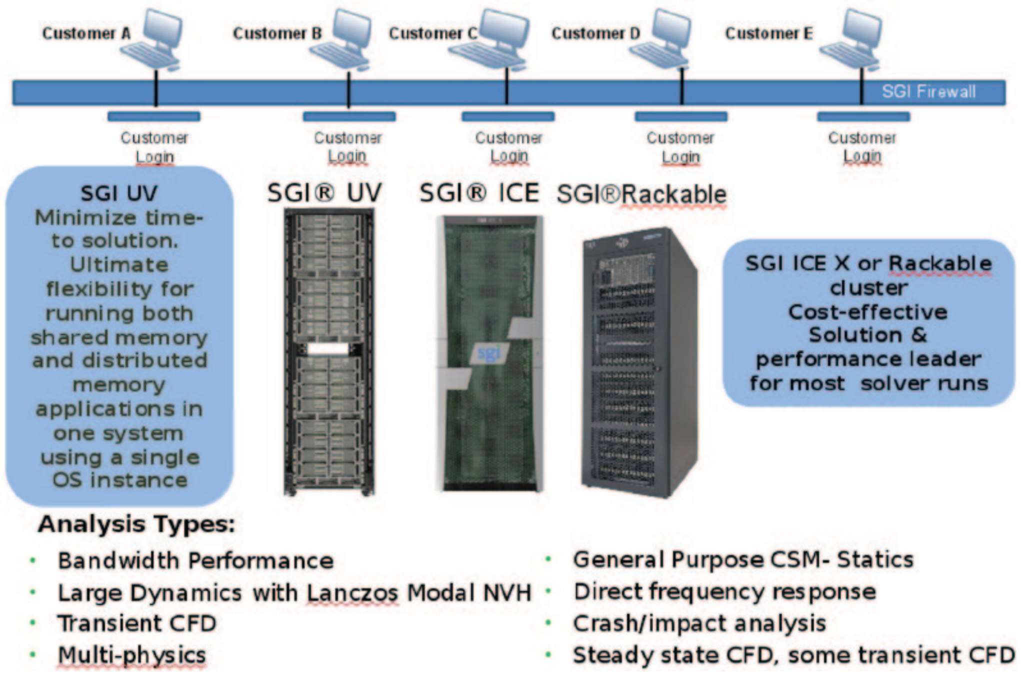
Customer A
Customer B
Customer C
Customer D
Customer E
SGI Firewall
Customer Login
Customer Login
Customer Login
Customer Login
Customer Login
SGI UV Minimize time-to solution. Ultimate flexibility for running both shared memory and distributed memory applications in one system using a single OS instance
SGI® UV
SGI® ICE
SGI®Rackable
SGI ICE X or Rackable cluster Cost-effective Solution & performance leader for most solver runs
Analysis Types:
Bandwidth Performance
Large Dynamics with Lanczos Modal NVH
Transient CFD
Multi-physics
General Purpose CSM- Statics
Direct frequency response
Crash/impact analysis
Steady state CFD, some transient CFD

Figure 5.2: SGI Cyclone Solution

2.0 MSC Nastran

2.1 Parallel Processing Capabilities of MSC Nastran

2.1.1 Hardware and Software nomenclature

Specific terminology is adopted differentiating processors and cores in hardware:

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

On the software side one distinguishes between:

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

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

2.1.2 Parallelism background

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

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

The two methods can be combined together in what is called 'Hybrid Mode'.

It has to be noted that while Shared Memory Processing cannot span cluster nodes both communication and memory-wise, Distributed Memory Parallelism can also be used within a Shared Memory system. Since DMP has coarser granularity than SMP, it is preferable, when possible to run DMP within Shared Memory systems (Liao, et al., 2007).

2.1.3 Parallelism metrics

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

(1) 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 is more dependent on CPU speed, and the serial part, comprising overhead tasks more dependent on RAM speed or I/O bandwidth. Therefore, a higher CPU speed system will have a larger 1-P serial part and a smaller P parallel part causing the Amdahl Speedup to decrease. This can lead to misleading assessment of different hardware configurations as shown by this example:

N	System A elapsed seconds	System B elapsed seconds
1	1000	640
10	100	80
Speedup	10	8

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

Speedup	10	12.5

Two other useful notions used for ranking supercomputers especially are:

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

It may be preferable, in the end, to instead use a throughput metric, especially if several jobs are running simultaneously on a system (2):

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

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

3.0 Tuning

3.1 Filesystem

An application involving I/O's relies on a high performance filesystem for best performance. This can be in-memory filesystem (/dev/shm), a Direct (DAS) or 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. This filesystem nomenclature is illustrated in Figure 6.

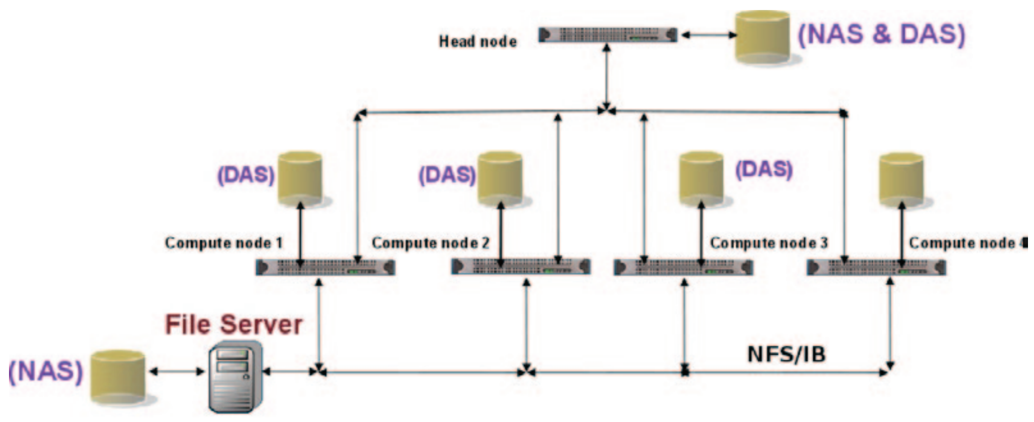

Figure 6: Example filesystems for scratch space

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.

Batch schedulers/resource managers dispatch jobs from a front-end login node to be executed on one or more compute nodes. 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

To achieve the best runtime in a batch environment input and output files should be placed on the high performance filesystem closest to the compute node. 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 input, auxiliary, working and output files between launch and the other nodes at start and end of the main analysis execution.

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 cost sizing blocks or through cores available which are their throughput sizing factors. When choosing the former, because processors have different prices, clock rates, core counts and memory bandwidth, optimizing for turnaround time or throughput will depend on running on all or a subset of cores available. Since licensing charges are assessed by the number of threads or processes being run as opposed to the actual number of physical cores present on the system, there is no licensing cost downside in not using all cores available. The deployment of threads or processes across partially used nodes should be done carefully in consideration of the existence of shared resources among cores.

3.3 Hyperthreading

Beyond 2 nodes, hyperthreading gains are negated by added communication costs between the doubled numbers of MPI processes. These results are not shown in this guide.

3.4 MPI tasks and OpenMP thread allocation across nodes and cores

For MSC Nastran, the deployment of processes, threads and associated memory is achieved with smp= and dmp= keywords in execution command.

3.5 SGI MPI, PerfBoost

The ability to bind an MPI rank to a processor core is key to control performance because of the multiple node/socket/core environments. From (Yih-Yih Lin et al., 2009), '3.1.2 Computation cost-effects of CPU affinity and core placement [...]HP-MPI currently provides CPU-affinity and core-placement capabilities to bind an MPI rank to a core in the processor from which the MPI rank is issued. Children threads, including SMP threads, can also be bound to a core in the same processor, but not to a different processor; additionally, core placement for SMP threads is by system default and cannot be explicitly controlled by users.[...]'. In contrast, SGI MPI, through the omplace command uniquely provides convenient placement of Hybrid MPI processes/OpenMP threads and Pthreads within each node. This MPI library is linklessly available through SGI MPI PerfBoost facility bundled with SGI Performance Suite. PerfBoost provides a Platform-MPI, IntelMPI, OpenMPI, HP-MPI ABI-compatible interface to SGI MPI.

3.6 SGI Accelerate libFFIO

MSC Nastran Explicit Nonlinear (SOL 700) is not I/O intensive, therefore, libFFIO is not necessary.

4.0 Software Environment

MSC Nastran is built on all SGI supported platforms, software stacks:

- x86-64, UV platforms, Single, Double Precision
- NUMA, multi-core, DMP, SMP
- Linux Distributions RHEL, SLES
- IntelMPI, Platform-MPI, OpenMPI
- MSC Nastran 2010.2 release started to include SGI Math kernels licensed by MSC Software.
- Distributed Memory Parallelism (DMP): Finite Element level using MPI API was introduced with v2001
- Shared Memory Parallelism (SMP): Numerical level using OpenMP API was introduced with v68

5.0 Benchmarks description

In the following sections, the models have been selected for increasing size and changes in solver, material, site and phenomenon modeled as indicated by their names.

5.1 Small model 'EulerSnowBoxCompression'

Model EulerSnowBoxCompression comprises 240k elements. Soil compressed by structure to test snow yield model, (Figure 7)

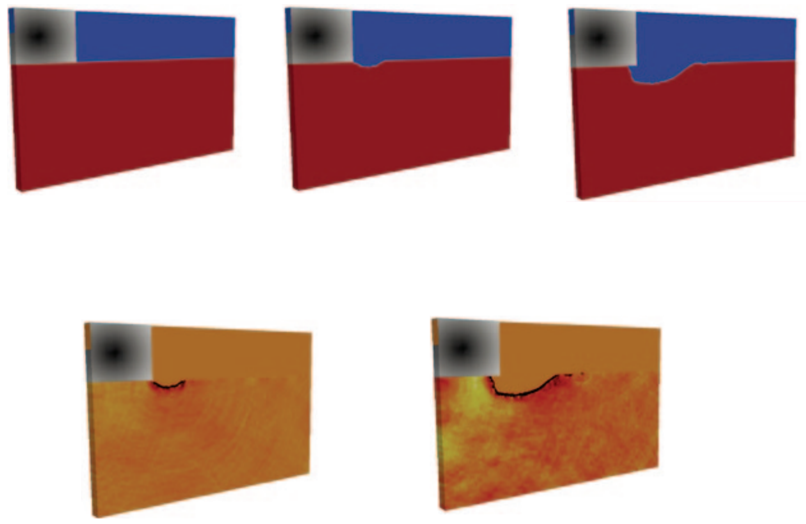

Figure 7: 'EulerSnowBoxCompression' model

5.2 Medium model ‘RoeAirBunkerBlast’

Model RoeAirBunkerBlast comprises 880k elements. Explosive is ignited outside bunker, blast wave hits bunker, bunker fails, explosive gas enters bunker. Flow between Euler domains is modelled. Uses second order Roe solver (Figure 8).

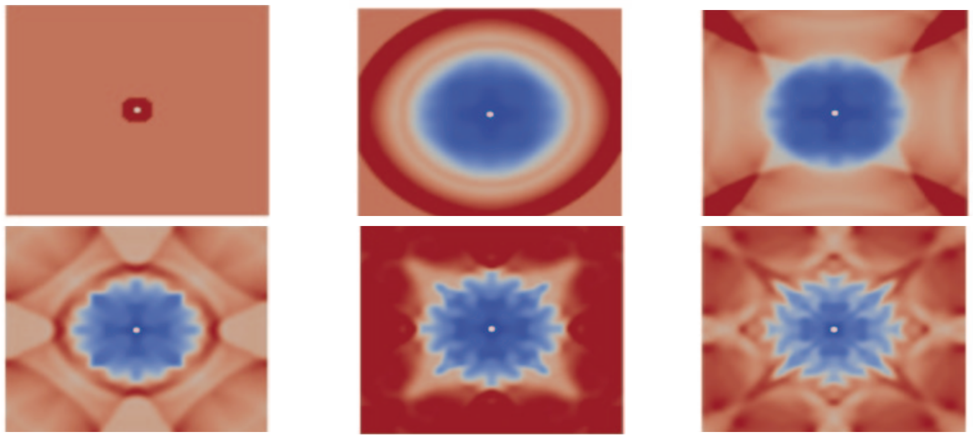

Figure 8: ‘RoeAirBunkerBlast’ model

5.3 Large model ‘RoeAirMineBlast’

Model RoeAirMineBlast comprises 2 Euler 800k elements sets. A blast wave expands below car model with opening, gas flows inside car. Flow between Euler domains is simulated. Uses the second order Roe solver (Figure 9).

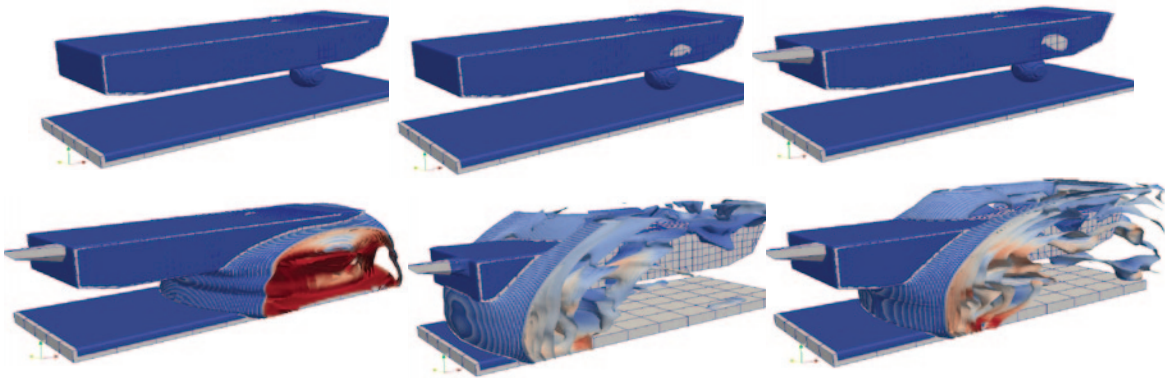

Figure 9: ‘RoeAirMineBlast’ model

5.4 Largest model 'EulerSoilVehicleBlast'

Model EulerSoilVehicleBlast comprises 2.2m elements to simulate soil covered JWL explosive below a vehicle. Uses biased Euler mesh. Blast location volume has especially fine Euler mesh to capture blast in soil. (Figure 10)

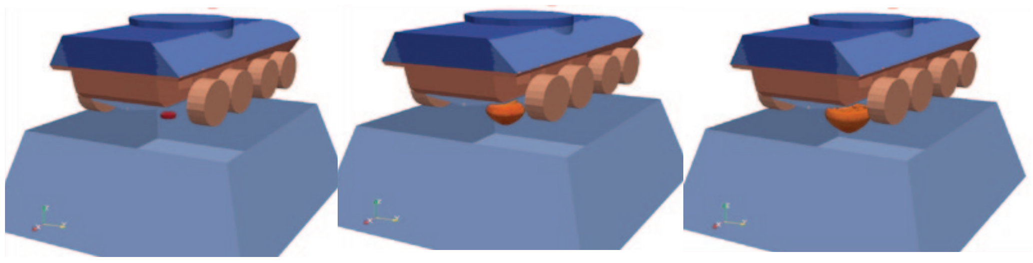

Figure 10: `EulerSoilVehicleBlast' model

6.0 Benchmarks Results

Several types of charts are used. The first type such as Figure 11 highlights the slower end of the performance curve by plotting elapsed times in seconds versus number of processes. The categories are the number of nodes used for each run. The second type such as Figure 12 highlights the faster end of performance curve by plotting number of jobs per day versus number of processes. The categories listed in the legend are the number of nodes used for each specific number of processes runs. For example, a column with abscissa of 64 and category of 8 means that 8 nodes were used evenly to run 64 processes, therefore using 8 cores per node.

6.1 CPU frequency effect on performance

For the example of case 5.1 one can see on both ends of the range that the ratio of 1.11 of frequency increase between 2.6GHz and 2.9GHz does not translate into an equivalent decrease of elapsed time for 16 processes (Figure 11) or an increase of job rating for 128 processes (Figure 12). The actual ratio average 1.05 showing that performance does not depend only on CPU frequency.

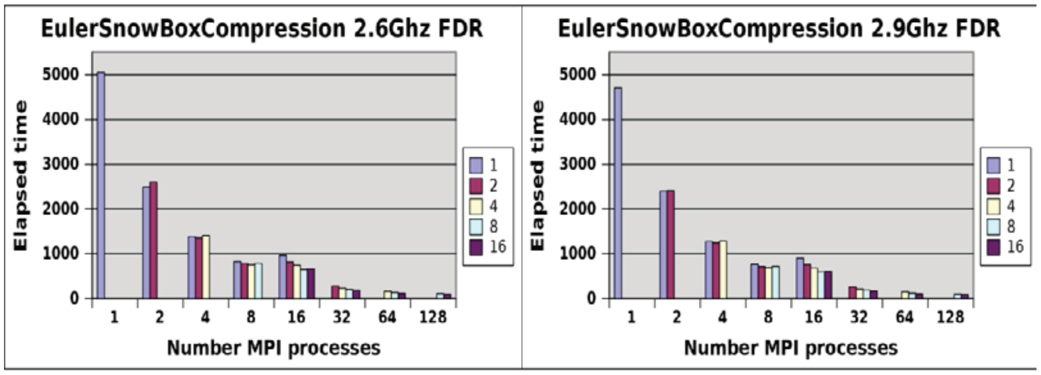

Figure 11: Comparison between 2.6 and 2.9GHz frequency for elapsed times

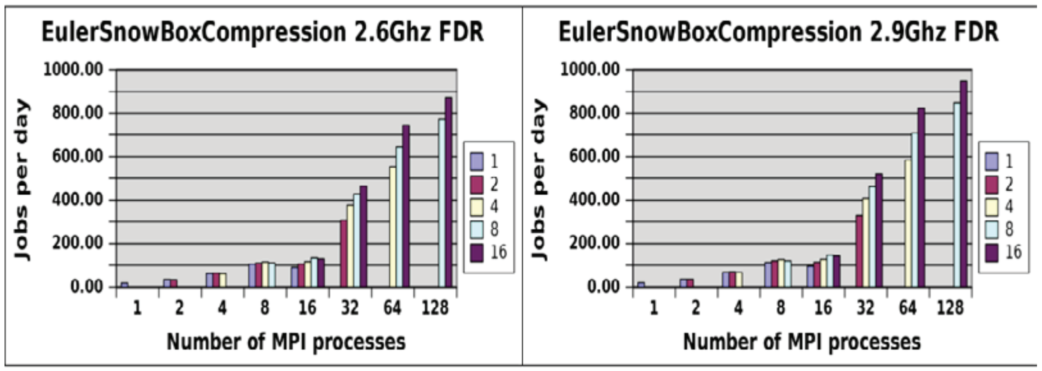

Figure 12: Comparison between 2.6 and 2.9GHz frequency for job ratings

6.2 Turbo Boost effect on performance

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 constrained by the thermal envelope. The mode of activation is a function of how many cores are active at a given moment when MPI processes or OpenMP threads or Pthreads are running or idle under their running parent. For example, for a base frequency of 2.93GHz, 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 most computation tasks, utilizing Turbo Boost technology can result in improved runtimes so it is best to leave it enabled although overall benefit may be mitigated by the presence of other performance bottlenecks besides pure arithmetic processing.

For case 5.1, both elapsed times for the low node counts (Figure 13) and job ratings in the higher values (Figure 14) show noticeable gains.

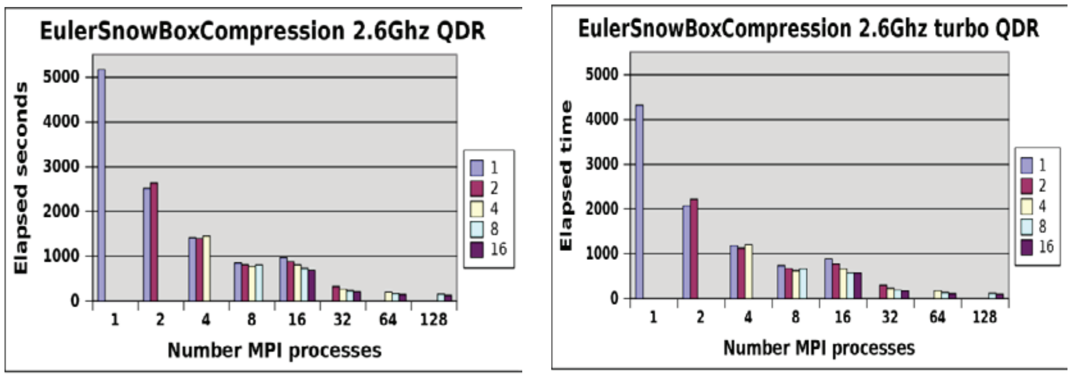

Figure 13: Comparison 2.6 versus 2.6GHz with Turbo Boost elapsed times

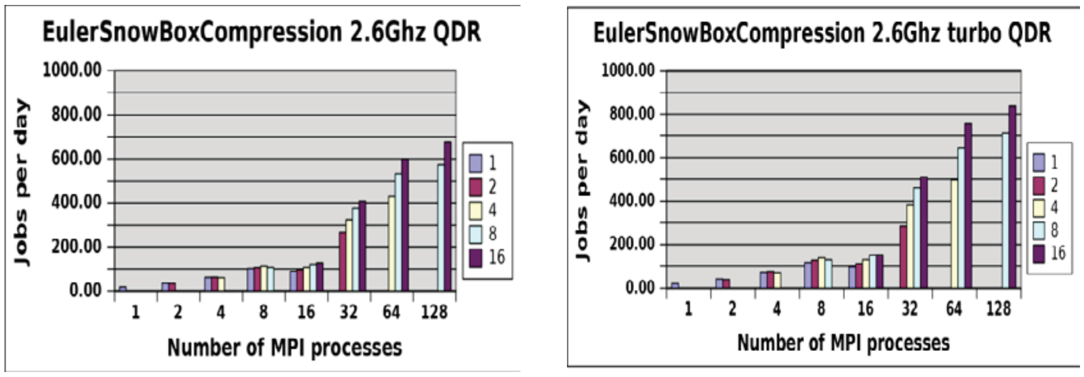

Figure 14: Comparison 2.6 versus 2.6GHz with Turbo Boost job ratings

6.3 Running on more nodes

Number of processes is the relevant parameter. Case 5.2 shows scaling continuing up 256 processes whereas for case 5.3 job rating falls above 128 processes (Figure 15)

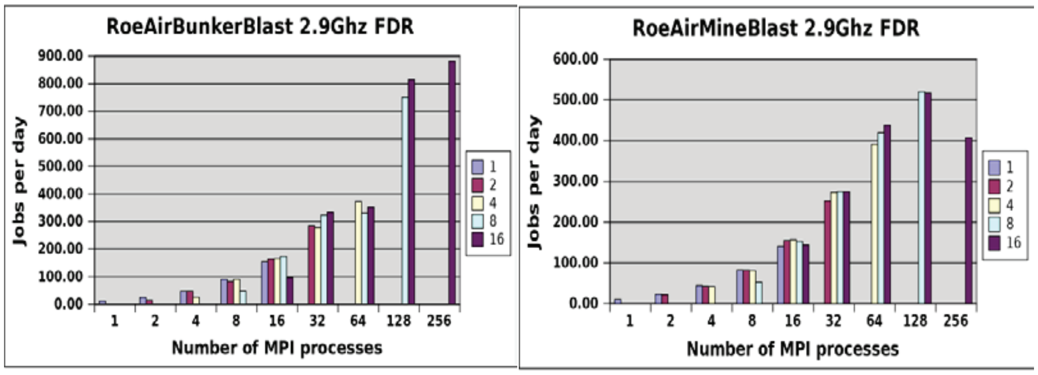

Figure 15: Scalability comparison between medium and large case at 2.9GHz frequency with FDR displayed by job ratings

6.4 Interconnect effect on performance

Case 5.1 shows performance gain going from QDR to FDR Infiniband interconnect (Figure 16) whereas case 5.4 does not show any performance gain going from QDR to FDR Infiniband interconnect (Figure 17)

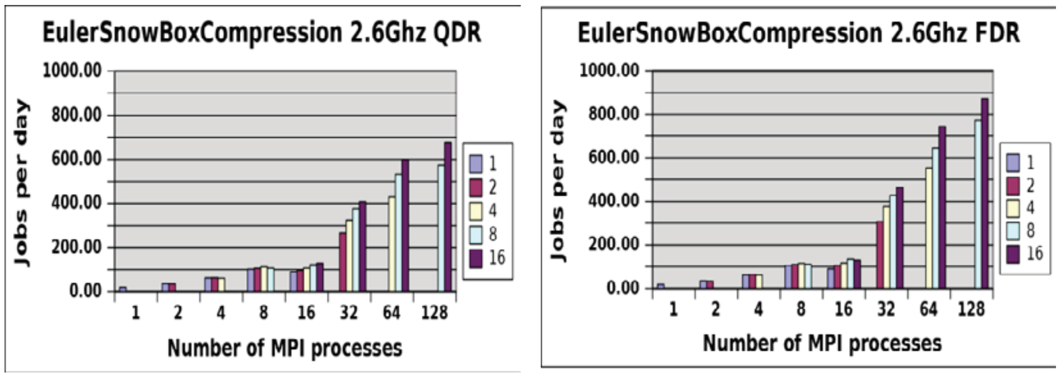

Figure 16: QDR vs. FDR Interconnect comparison for small case at 2.6GHz frequency displayed by job ratings

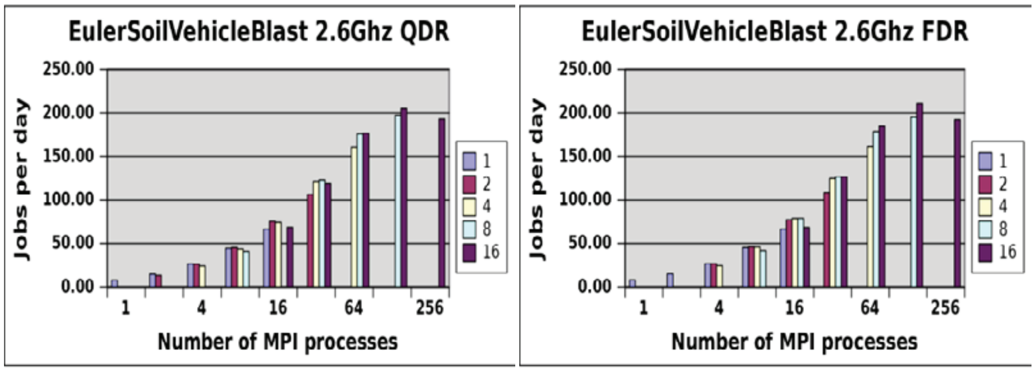

Figure 17: QDR vs. FDR Interconnect comparison for largest case at 2.6GHz frequency displayed by job ratings

6.5 Standard performance interconnect (GigE) vs QDR interconnect

Lack of scalability past 2 nodes is shown by Figure 18. When the number of licenses or nodes is limited then this may not be an issue.

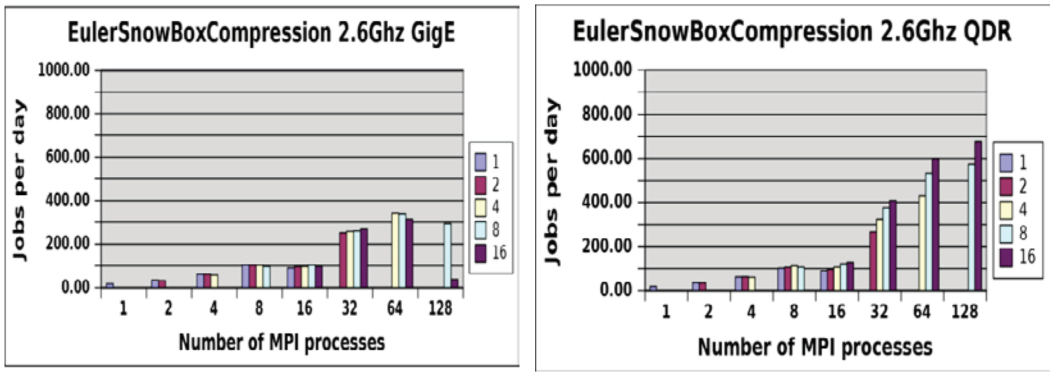

Figure 18: GigE vs. QDR Interconnect comparison for small case at 2.6GHz frequency displayed by job ratings

6.6 Leaving cores unused across nodes

For the small case 5.1 when turnaround time maximization is desired, spreading out processes on extra available nodes and leaving some cores unused can bring about increased performance whereas the medium case 5.2 doesn't show any worthwhile benefit using that same strategy (Figure 19).

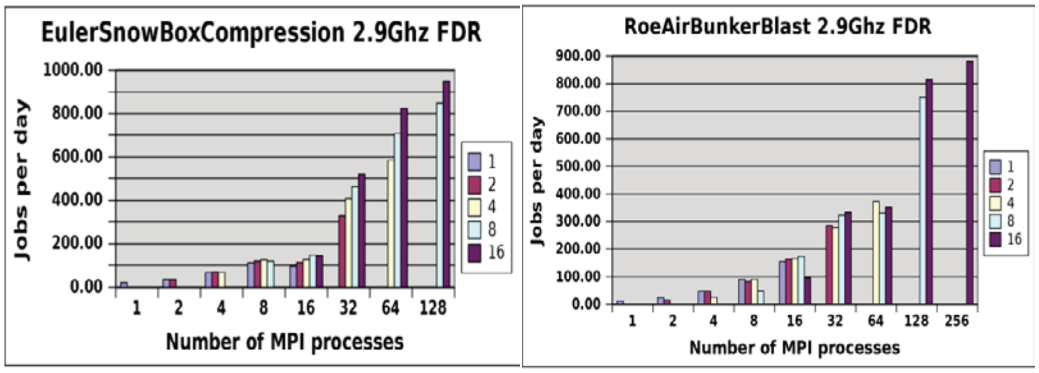

Figure 19: Benefit of spreading out processes across nodes small case on the left figure and no benefit for medium case on the right at 2.9GHz frequency FDR as displayed by job ratings

MPInside, an MPI profiling and performance analysis tool that provides finer-grained metrics for analysing MPI communications (Thomas, 2011) was used to separate timings imputed to computational work or communications. Computational work is the bottom blue layer. MPInside profile of the computation and communication times spent by the 64 processes runs for case 5.1 (Figure 20.1, 20.2, 20.3) and 5.2 (Figure 21.1, 21.2, 21.3) illustrates how an underlying load balance problem may cause the decrease in performance in the latter. The opposite concept is exemplified by Hyper-Threading, (HT) which is a feature that can increase performance for multi-threaded or multi-process applications. It allows a user to oversubscribe each node's physical cores by running two OpenMP threads or MPI processes on each of them. No benefit was found using hyperthreading in this study.

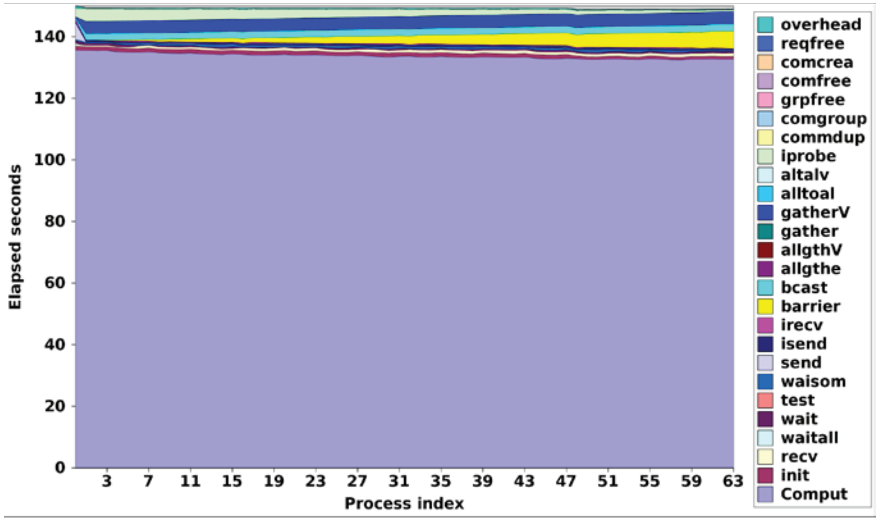

Figure 20.1: 64 processes run across 4 nodes, small case at 2.9GHz frequency FDR displayed by computation and communication costs

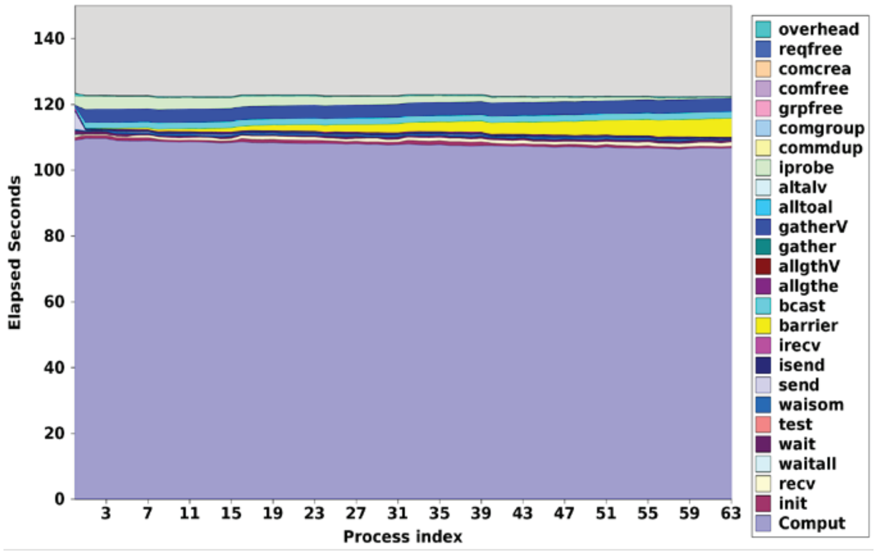

Figure 20.2: 64 processes run across 8 nodes, small case at 2.9GHz frequency FDR displayed by computation and communication costs

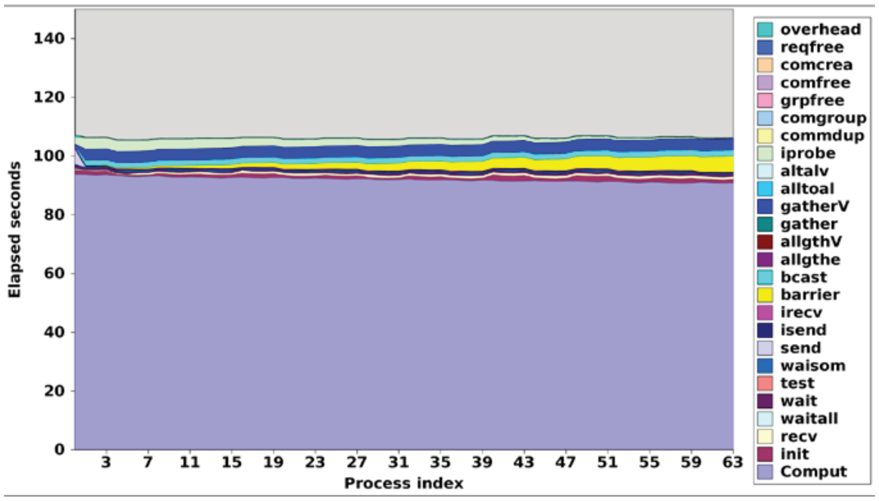

Figure 20.3: 64 processes run across 16 nodes, small case at 2.9GHz frequency FDR displayed by computation and communication costs

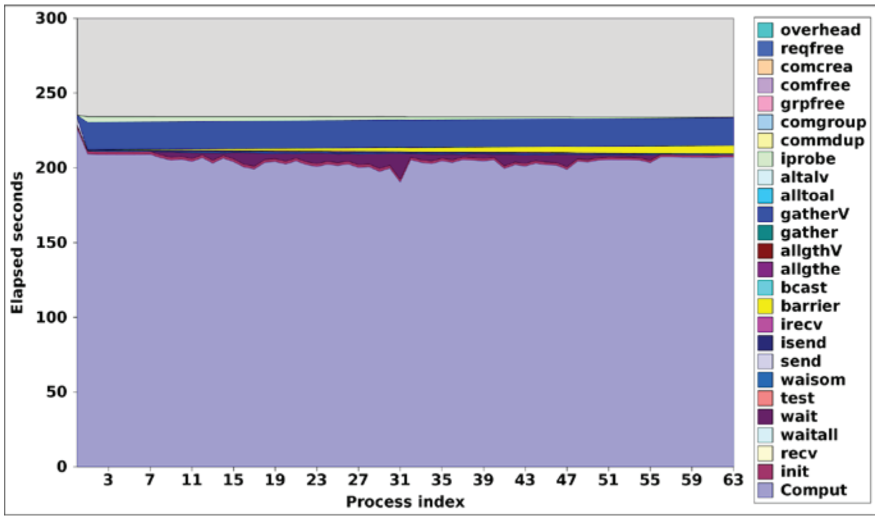

Figure 21.1: 64 processes run across 4 nodes, medium case at 2.9GHz frequency FDR displayed by computation and communication costs

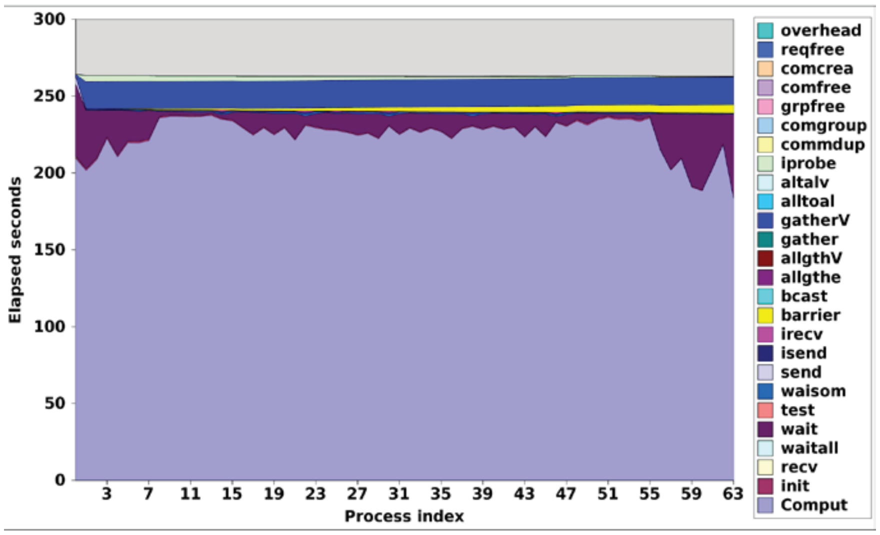

Figure 21.2: 64 processes run across 8 nodes, medium case at 2.9GHz frequency FDR displayed by computation and communication costs

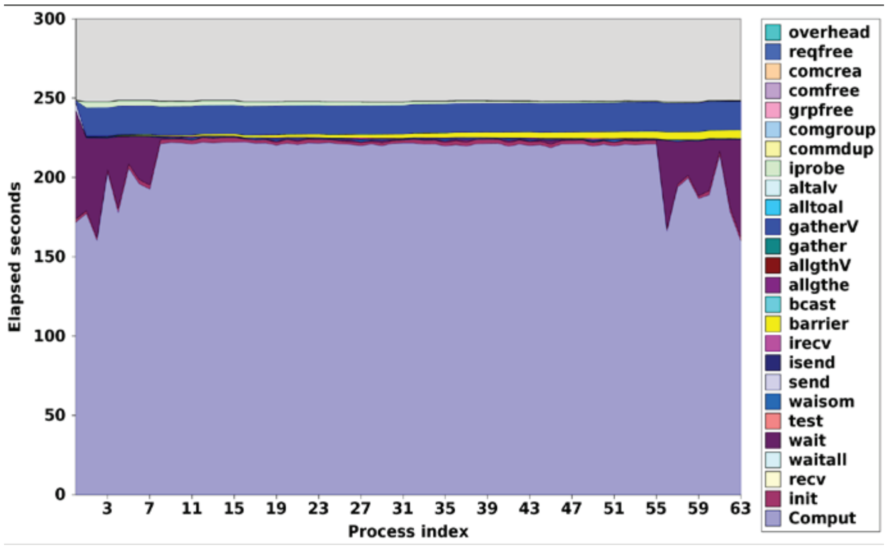

Figure 21.3: 64 processes run across 16 nodes, medium case at 2.9GHz frequency FDR displayed by computation and communication costs

6.7 Anomalies

Figure 22 shows an anomaly for case 5.2 where 2 processes on 2 nodes take 60% more elapsed times than on 1 node. MPInside charts (Figure 23) show the appearance of wait time caused by load unbalance. Fortunately, this anomaly would not be seen, since, at a minimum, 16 processes would be run on these datasets.

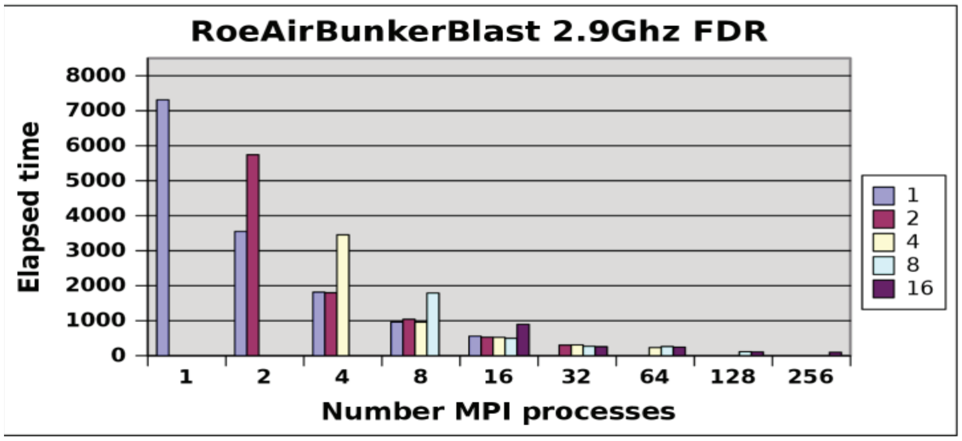

Figure 22: Anomaly for medium case of 2 processes on two nodes vs. one node

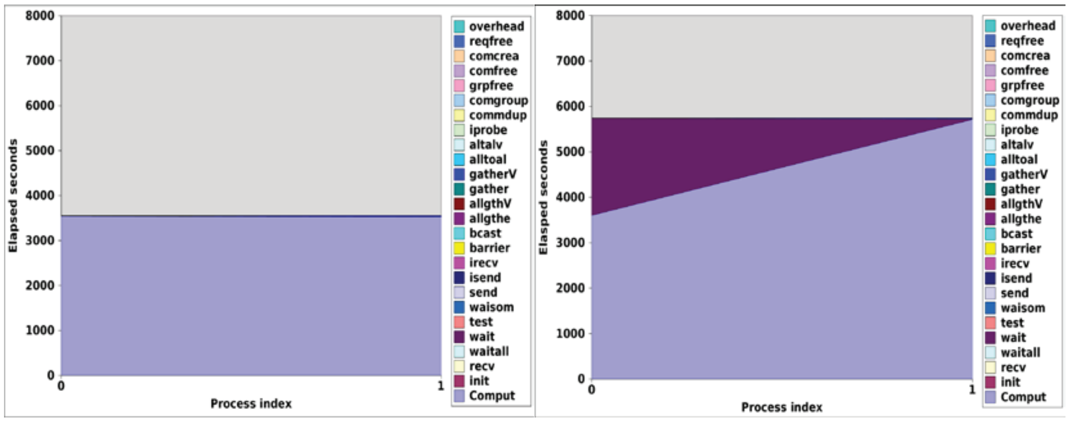

Figure 23: Anomaly for medium case of 2 processes on two nodes vs. one node

7 Summary

MSC Nastran Explicit Nonlinear (SOL 700) offers integrated shared and distributed parallel computational features applicable to SGI advanced computer architectures, which, combined with advanced interconnect and associated libraries, core frequency, Turbo Boost, allows greatly enhanced workflow optimization possibilities. Effects on performance can be gauged for a given dataset. In particular one observed:

- Infiniband interconnect can allow scaling and be twice as fast as GigE.
- Effect of Frequency and Turbo Boost are weak as CPU is not the only limiting factor for performance.
- HyperThreading does not increase performance because of communication costs beyond 16 processes.

All these effects are seen to be dependent on datasets. Procurement of the right mix of resources should therefore be tailored to the mix of datasets envisaged. Upgrading a single system attribute like CPU frequency, interconnect, number of cores per node, RAM speed, brings diminishing returns if the others are kept unchanged. Trades can be made based on metrics such as dataset turnaround times or throughput, acquisition, licensing, energy, facilities, maintenance costs to minimize.

8 Attributions

MSC Nastran is a registered trademark of MSC Software Corporation. LS-DYNA is a registered trademark of Livermore Software Technology Corp. SGI, ProPack and Cyclone are registered trademarks or trademarks of Silicon Graphics International Corp. or its subsidiaries in the United States or other countries. Xeon is a trademark or registered 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.

9 References

SGI (2009), Linux c Application Tuning Guide. Silicon Graphics International, Fremont, California.

C. Liao, O. Schreiber, S. Kodiyalam, and L. Lewis, (2007) “Solving Large Problems on Large Computer Systems. Hardware Considerations for Engineering Productivity Gains with Multi-Discipline MSC NASTRAN Simulations ”. In MSC Software R VPD Conference, October 11-12 2007.

Yih-Yih Lin and Jason Wang, (2009). “Performance of the Hybrid LS-DYNA on Crash Simulation with the Multicore Architecture”. In 7th European LS-DYNA Conference, 2009.

Daniel Thomas, (2011), `A Performance Analysis and Diagnostic Tool for MPI Applications’,, http://www.sgi.com/global/fr/pdfs/LB_SGI_MPInside.pdf

Global Sales and Support: sgi.com/global