WHITE PAPER

NX NASTRAN™ on Advanced SGI® Architectures*

Olivier Schreiber †, Scott Shaw †, Louis Komzsik**

Abstract

NX NASTRAN tackles all important Normal Mode Analysis (NMA) utilizing both Shared Memory Parallelism (SMP) and Distributed Memory Parallelism (DMP). The technical approaches include frequency and geometry domain decompositions and their hierarchical and recursive executions. Efficient execution of above four solutions and two paradigms requires extreme care in allocating hardware resources. The topic is even more important given the hardware variety available today, ranging from single node multi-core workstations through multiple node clusters to single system image many-core systems addressing a very large memory space. The paper will explore memory, input/output, communication latency and bandwidth requirements of such solutions to establish guidelines for advanced SGI computer hardware systems.

** Adapted from [1]*
† SGI Applications Engineering
*** Siemens PLM, Office of Architecture and Technology*

TABLE OF CONTENTS

Introduction

New hardware such as SGI Octane™ III, SGI Altix® XE, SGI Altix UV architectures are available (Figure 1) to extend computational performance. NX NASTRAN computational technologies such as Shared Memory Parallel (SMP), Distributed Memory Parallel (DMP) and their combination (hybrid mode) can exploit this advanced hardware for normal modes analyses. The strategies employed by NX NASTRAN in Refs [2] and the practical importance of such analyses is demonstrated in Ref [3]. How to best use SGI hardware is described in Ref [4]

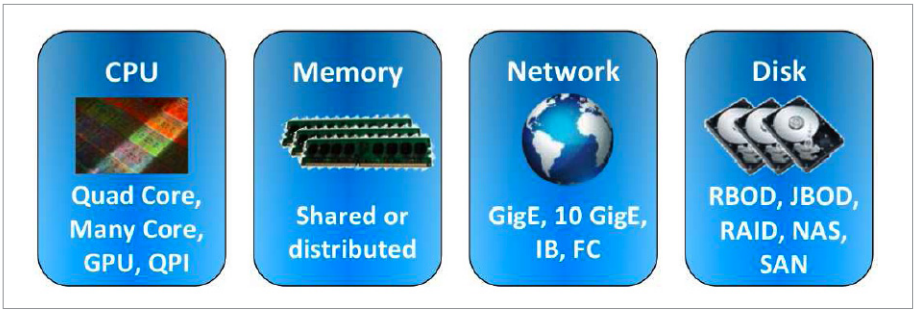

Figure 1: New hardware technologies

1. SGI hardware overview

Various systems comprised in SGI product line and available through SGI Cyclone™ , HPC on-demand Cloud Computing (see section 1.5) were used to run the benchmark described in section 3.1.

1.1. SGI Octane III Xeon® X5570

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 quad-core Xeon X5570, 8 MB cache.
- Total Memory: 24 GB Speed: 1333 MHz (0.8 ns)

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.

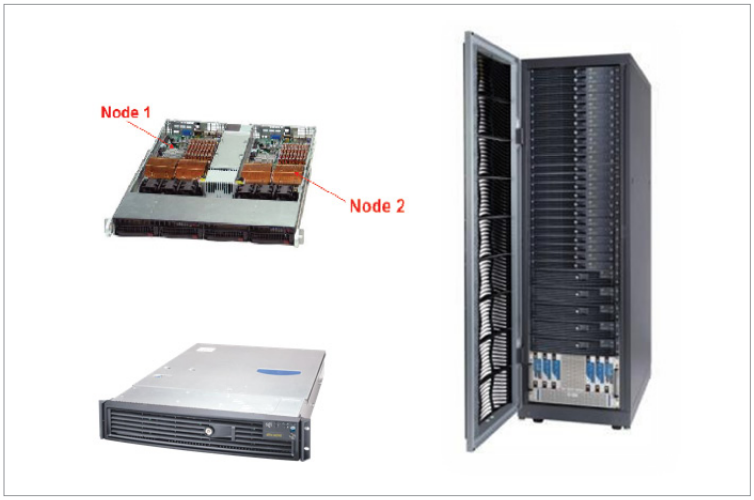

Figure 3: SGI Altix XE 1300 cluster

- SGI XE250 Administrative/NFS Server node
- SGI XE250 Dual-socket compute nodes of 3.0GHz quad core Xeon X5472 12MB Cache, 1600MHz Front Size Bus.
- 32GB 1333MHz RAM
- 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.

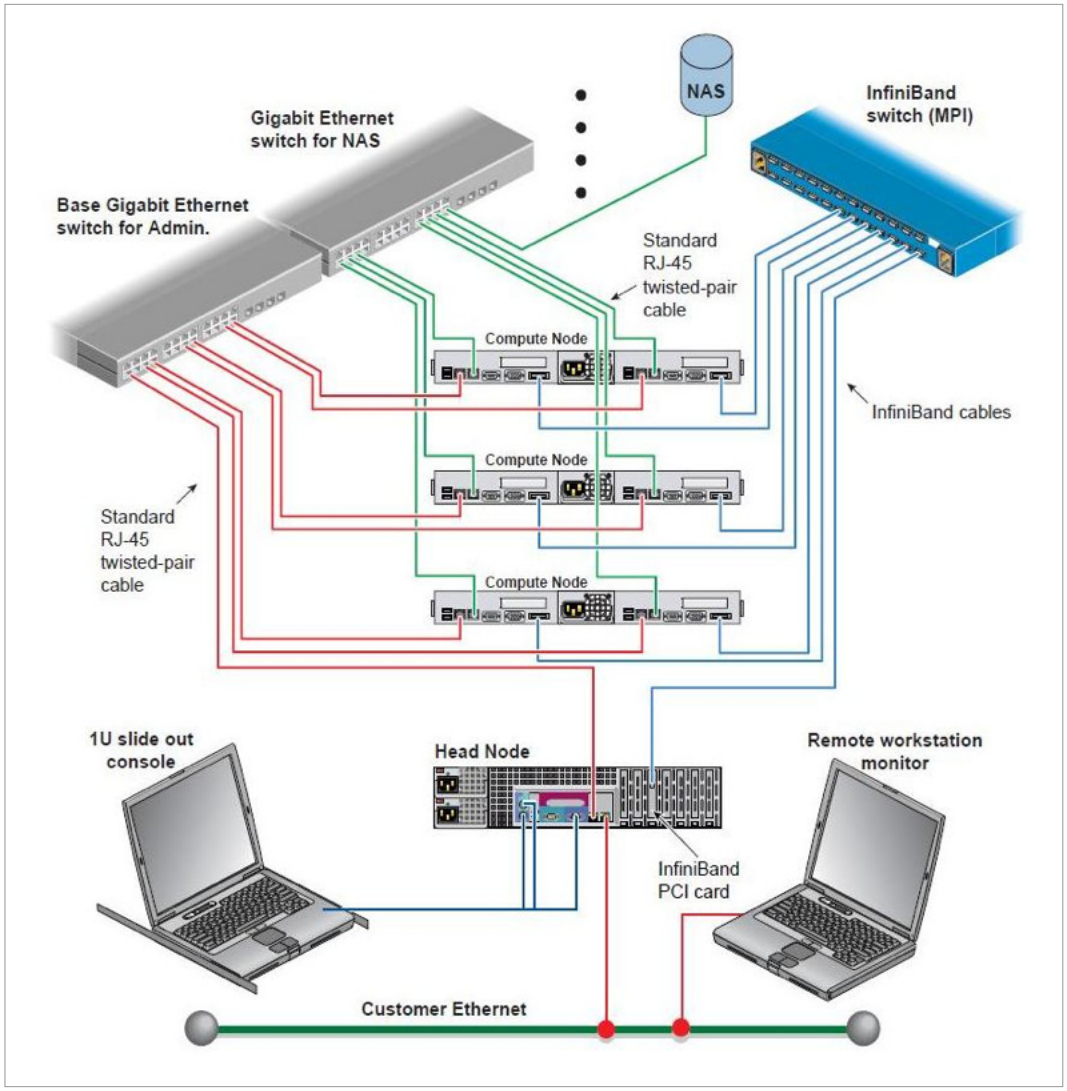

Figure 4: Dual Ethernet and Infiniband switch cluster configuration example

1.3 SGI Altix UV 100, UV 1000 SMP

Highly scalable latest generation x86-based Shared Memory Parallel system.

Figure 5: SGI Altix UV 10, UV 100, UV 1000 SMP

- 12 2.66Ghz 6-core Xeon X7542 (72 cores)
- 192GB RAM
- SGI® NUMAlink® 5
- SGI Foundation Software, SGI ProPack 7

1.4 Advanced SGI I/O systems

NX NASTRAN is an I/O dominated application which 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. In cluster computing environments with a common scratch location, isolating application MPI communications and NFS traffic will provide the best NFS I/O throughput for scratch files. Various file systems can be setup to use for scratch space:

- Memory-based filesystem /dev/shm
- Root drive
- DAS (Direct Attached Storage)
- NAS (Network Attached Storage)

This file system nomenclature is illustrated in Figure 6

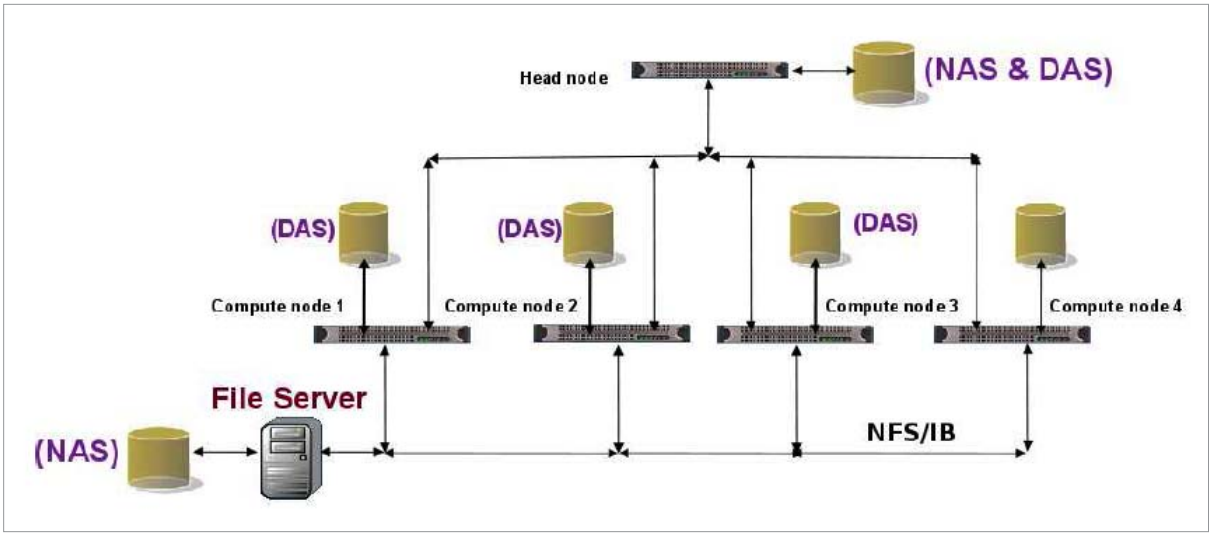

Figure 6: Example file systems for scratch space

Another factor regarding performance is memory per core. Having more memory per core in most cases will increase the performance of NX NASTRAN since more memory can be allocated for the analysis and any unallocated memory will be used by the Linux kernel buffer cache.

SGI's Flexible File IO (FFIO) library can help to minimize buffer cache contention when running multiple NX NASTRAN jobs in Shared Memory Parallel systems or cluster computing environments using DAS or NAS storage subsystems

Indeed, a major benefit of SGI's FFIO library is the ability to minimize the number of system calls and I/O operations to/from the storage subsystem. Regardless of running a single job or multiple jobs per node, SGI's FFIO library will always produce faster throughput of I/O operations and as a direct result NX NASTRAN computational steps will be more efficient (Ref [4], Chapter 7 Flexible File I/O.)

1.5 Cloud access to benchmark systems: SGI Cyclone™

SGI offers Cyclone, HPC on-demand computing resources of all SGI advanced architectures aforementioned. There are two service models in Cyclone. Software as a Service (SaaS) and Infrastructure as a Service (IaaS). 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 (ISVs).

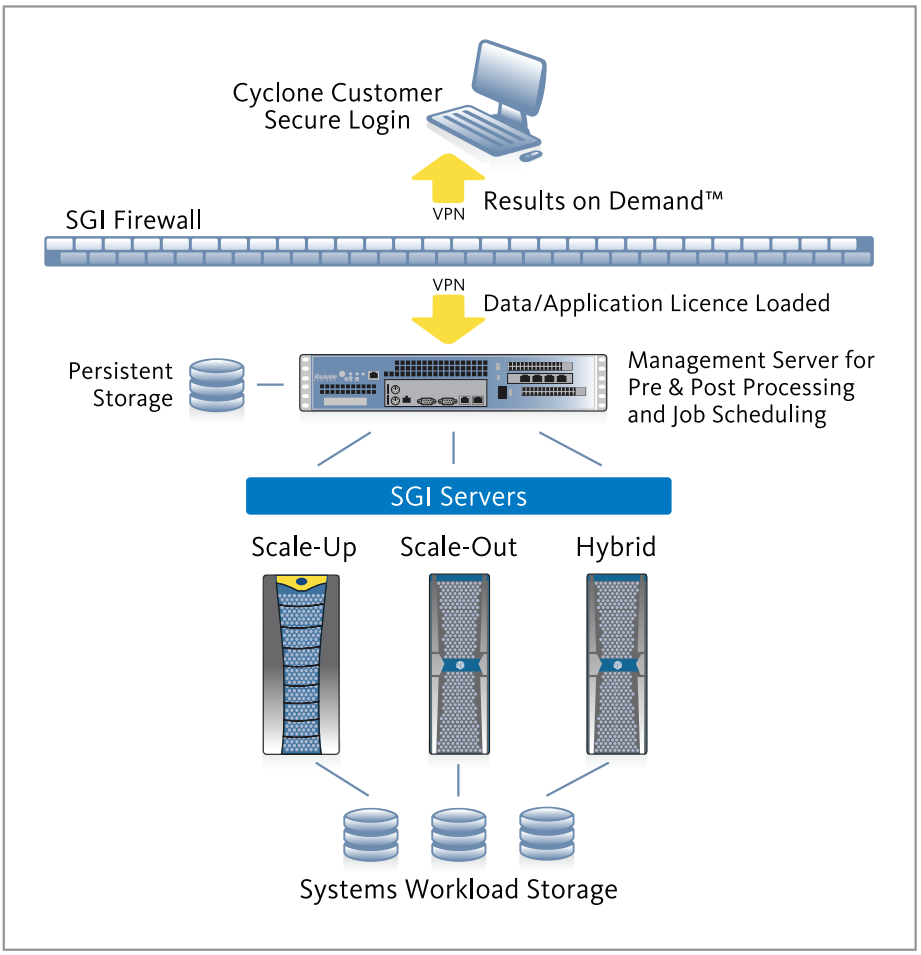

Figure 7: SGI Cyclone – HPC on-demand Cloud Computing

2. NX NASTRAN Overview

2.1 Parallel Processing Capabilities of NX NASTRAN

Parallelism in computer systems exists in two paradigms:

- Distributed Memory Parallelism (DMP) uses MPI Application Programming Interface focused on Finite Element computations or physical domain decomposition. The resulting reduced size geometric partitions have smaller processing resource requirements, resulting in increased efficiency, but the size of the common boundary should be kept minimal to decrease inter-process communication.

- Shared Memory Parallelism (SMP) uses OpenMP Application Programming Interface focused on numerical modules.

These two paradigms can simultaneously map themselves on two different system hardware levels:

- Inter-node or cluster parallelism (memory local to each node)–DMP only
- Intra-node or multi-core parallelism (memory shared by all cores of each node)

The hybrid approach provides increased performance yields and additional possibilities of memory utilization by running SMP on intra-node network simultaneously with DMP on inter-node and/or intra-node network. NX NASTRAN exploits parallelism through these two paradigms. For cluster computing with multi-core processors, NX NASTRAN can also run DMP for both inter-node and intra-node.

2.2 Distributed parallel capabilities in NX NASTRAN

These capabilities seek to partition the problem to decrease processing resource requirements while minimizing the size of common boundaries to decrease inter-process communication (figure 8). The partitioning is executed either in the Finite Element model or at the Matrix Level.

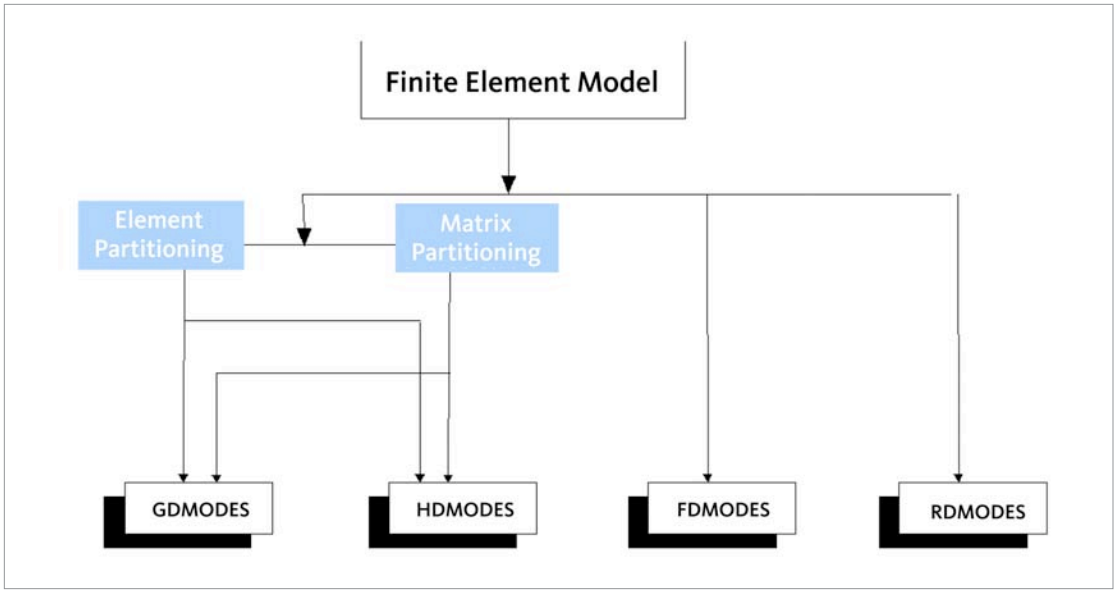

Figure 8: Parallel Options

2.2.1 Frequency domain decomposition: FDMODES

The FDMODES approach partitions the frequency range of the analysis tasks into segments of approximately equal number of natural frequencies and distributes them among the nodes/processors. Normal modes and dynamic response computations in the frequency segments are independent. (figure 9) Load vectors can also be partitioned and assigned to different nodes/processors.

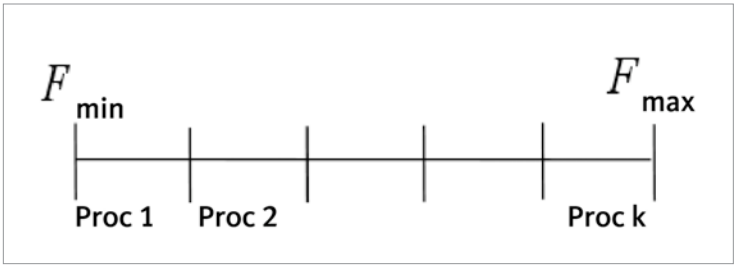

Figure 9: FDMODES

2.2.2 Geometry domain decomposition: GDMODES

The GDMODES technique partitions the geometry model (either the finite element graph or the finite element matrices) and distributes the partitions among the nodes/processors. The computations in the geometry partitions are dependent through their common boundaries and care must be taken of minimizing the boundaries sizes between partitions to minimize communication costs (figure 10).

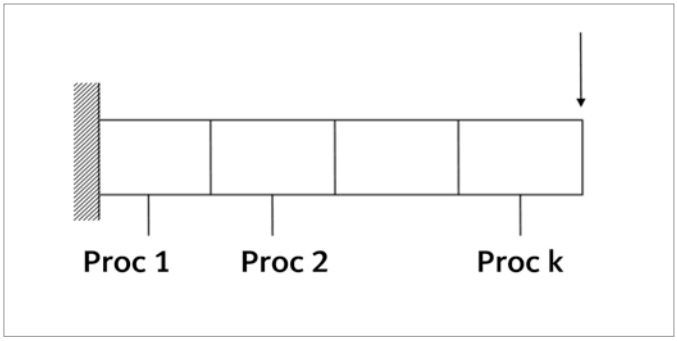

Figure 10: GDMODES

2.2.3 Hierarchic domain decomposition: HDMODES

The HDMODES is a hierarchic application of the FDMODES and GDMODES techniques by simultaneously dividing the frequency domain into segments and the geometry domain into partitions. This method is advantageous on hierarchical computer architectures. The number of MPI processes created is equal to the product of the number of geometric partitions and frequency segments (figure 11).

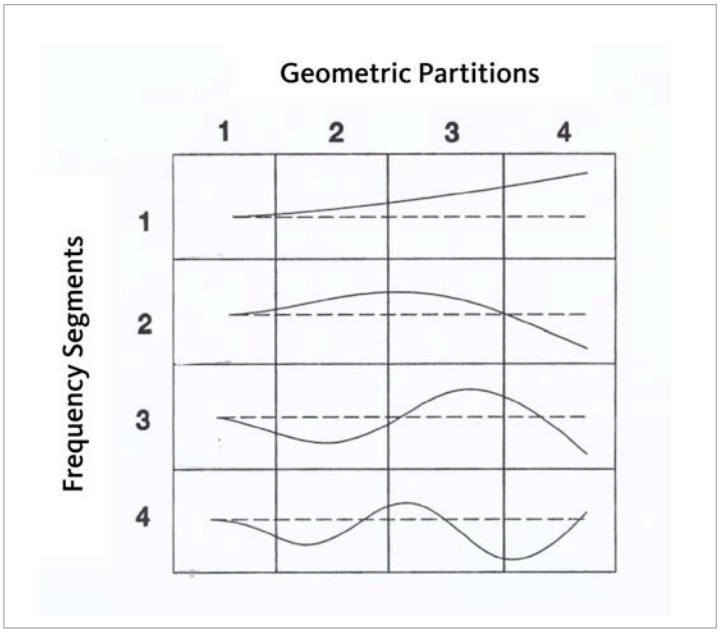

Figure 11: HDMODES

Figure 12 depicts the application of the HDMODES method with n geometric partitions and m frequency segments. The preferred hierarchical decomposition consists in partitioning geometry within each node and analyze over a distinct frequency segment across nodes to minimize communication traffic between nodes.

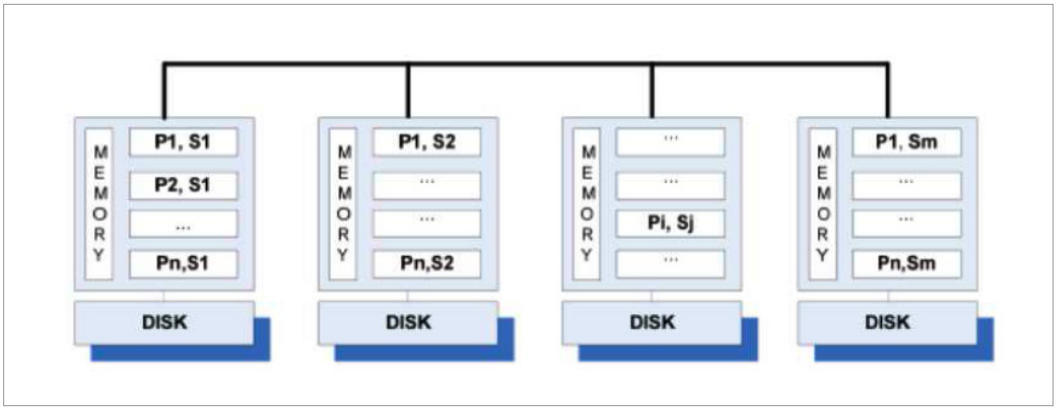

Figure 12: Mapping the hierarchic domain decomposition (Pi: geometric domain partition and Sj: frequency segment) to a cluster of multi-processor workstations

2.2.4 Recursive domain decomposition: RDMODES

The RDMODES is the highest level technique that recursively partitions the model into multilevel substructures (figure 13) resulting in a higher number of geometry partitions (parameter nrec) than the number of MPI processes created (parameter dmp).

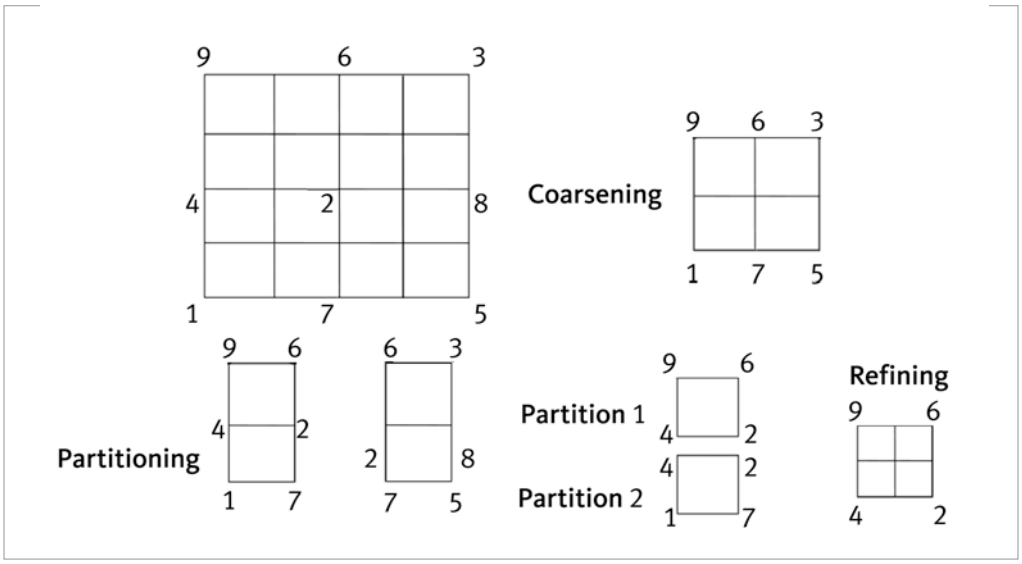

Figure 13: RDMODES coarsening, partitioning and refining phases

At various stages of this technique the GDMODES and FDMODES approaches are recursively applied. In all these schemes allocation of the total number of MPI processes over the nodes may be made by filling up each node's cores designated for processing first ('rank' allocation) or by distributing them in round-robin fashion across all the nodes.

2.3 Distributed execution control

2.3.1 Submittal procedure

Submittal procedure must ensure:

- Placement of processes and threads across nodes and cores 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 of course the only options.

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.

2.3.2 Submittal command

The following keywords were used for the NX NASTRAN execution command:

- **dmp:** Total number of MPI processes used in a Distributed Memory Parallel job.
- **nclust:** Number of nodes to be used in a hierarchic distributed memory parallel job.
- **hosts:** List of the nodes used in a Distributed Memory Parallel job. The ordering determines whether geometry and frequency range will be split or striped across nodes.
- **mem:** Size in words of allocated RAM for each MPI process.
- **sdir:** List of scratch directories allocated for each MPI process.
- **gpart:** Selects FE Model or Matrix Partitioning for the geometric domain decomposition.
- **nrec:** Selects number of geometric partitions for Recursive Domain Lanczos Method.

Various NX NASTRAN distributed solutions can be exercised in this manner (figure 14).

Analysis Solution	Modal Computation	Response Computation	Submittal Command
	GDMODES		dmp =p
Modal frequency response: SOL111 Modal transient response: SOL112	FDMODES	Distributed frequency Serial transient	dmp =p, numseg=p
Design optimization: SOL 200	HDMODES	Serial optimization	dmp =p, nclust=c
	RDMODES		dmp =p, nrec=n

Figure 14: NX NASTRAN distributed solution scenarios

3. Results

3.1 Benchmark example

The benchmark used is a car body model similar to Figure 15. It has approximately 268,000 Grids, 275,000 Elements, 1,584,000 degrees of freedom. The solution sequence performed is NX NASTRAN's Sol103 with all roots requested in the frequency range of 0 to 200Hz and 0 to 400Hz, which is approximately 1000 and 3000 modes, respectively.

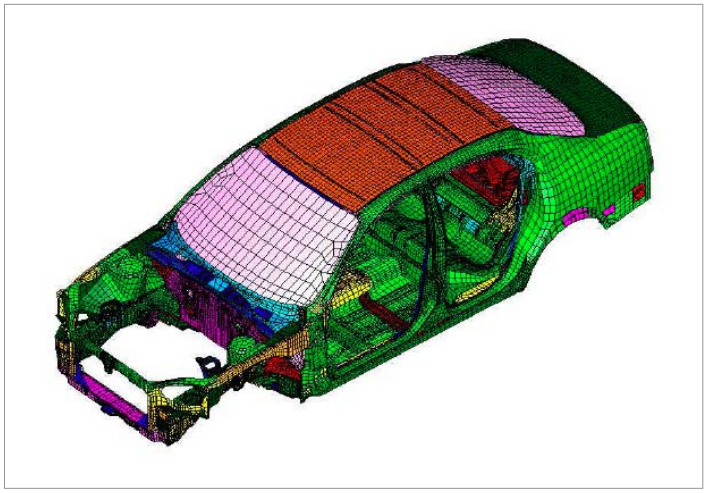

Figure 15: car body Model

3.2 Benchmark Results on SGI Altix XE 1300

3.2.1 Best Configurations

The first line of table 1 is the serial run. In this table and the following ones, P is the number of MPI processes per job, N is number of Nodes used per job, c is number of cores used per node, and times are job elapsed runtimes in seconds. Using 2 nodes, 8 MPI processes are optimum out of the 16 cores available. The best configuration found on table 1 is with the frequency range striped 2-way across the nodes and the geometry domain 4-way split across the nodes. Using 4 nodes, similarly, 16 MPI processes are optimum out of the 32 cores available. The best configuration found is with the frequency range striped 4-way across the nodes and the geometry domain 4-way split across the nodes.

P	N	c	times	Description
1	1	1	11772	Serial processing, 1 frequency band, 1 geometry domain
8	2	4	2053	frequency 2-way split, geometry 4-way striped across nodes
8	2	4	1912	frequency 2-way striped, geometry 4-way split across nodes
8	4	2	1595	frequency 4-way split/striped, geometry 2-way split/striped across nodes
16	4	4	1220	frequency 4-way split, geometry 4-way striped across nodes
16	4	4	1160	frequency 4-way striped, geometry 4-way split across nodes
32	4	8	1334	frequency 4-way striped, geometry 8-way split across nodes

Table 1: Best runs configurations (to 200Hz)

Thus, elapsed time was unexpectedly better on the run where the geometry was split and the frequency range was striped across the nodes than the other way around. The latter pattern would normally minimize communication traffic between nodes. In this benchmark, though, the number of eigenvalues computed in each frequency band was not perfectly uniform because heuristics are used to determine frequency partitioning. This results in a load imbalance between the nodes when the frequency range is split across them. It will be worthwhile, therefore, after inspection of the analysis output to flip the splitting and striping of frequency range and geometry every time a new dataset is studied.

3.2.2 HDMODES Results

Runs were performed by increasing the number of MPI processes per node from 1 to 8 as shown by figure 16. The maximum number of processes per node is 8 which is the number of cores existing on each node's pair of Xeon quad core processors. Reading figure 16, a run with P=32 in a column indexed by a number of nodes equal to 4 means 8 processes were run on each node, that is utilizing all 8 cores of each node. In another case, a run with P=2 in a column indexed by a number of nodes equal to 2 means only 1 process was run on each of the 2 nodes used for that job. For a number of nodes equal to 2 and 4, it is apparent that above 4 MPI processes per node, performance decreases because the memory bandwidth available per node becomes saturated. Hence, the optimum number of MPI processes per node in these cases is 4. On the other hand, an

unexpected good result is that for single node runs performance increases up to P=8 MPI processes on that single node. The explanation is that single node runs do not have inter-node communication costs and this makes up for the aforementioned lack of bandwidth when even 8 cores are used by 8 MPI processes.

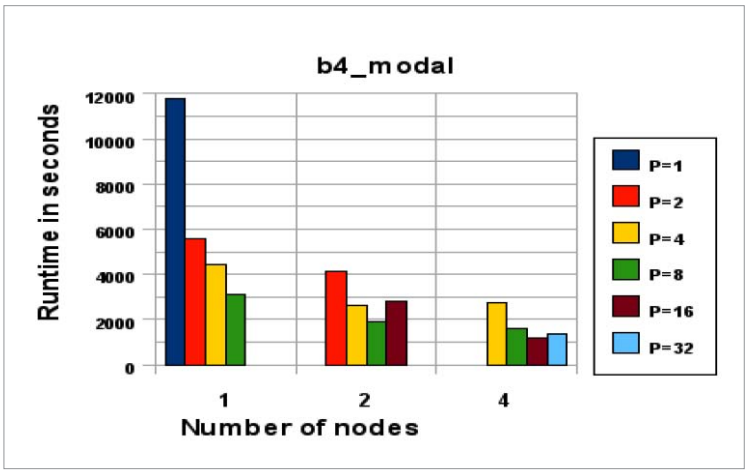

Figure 16: Benchmark runtimes by changing # of MPI processes per node (to 200Hz)

Since clusters are procured in terms of nodes for desired improvements in turn-around time, a more appropriate way to assess cluster performance is by plotting run-times against the number of nodes used. This can help for comparisons with other systems where a different number of MPI processes per node out of a different number of cores available were run. This is accomplished by taking a subset of the columns in figure 16 to figure 17 which shows the best run times obtained for the numbers of nodes brought on to process the job.

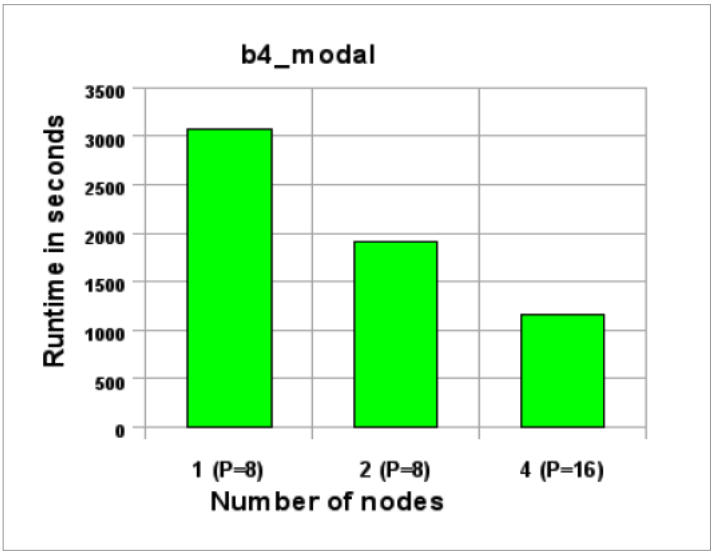

Figure 17: Benchmark results against node utilization (to 200Hz)

These results show that for a particular analysis job the optimum number of MPI processes resulting in the best performance depends on and varies with the number of nodes used.

3.2.3 RDMODES Results

Table 2 lists the best run configuration of 16 MPI processes on 4 nodes with Node Fill (RANK) and Round Robin (RR) schemes of MPI process allocation and various nrec values. The optimum run is with nrec=256 and Round-Robin allocation.

P	N	c	times	Description	allocation	nrec
16	4	4	604	RDMODES	RANK	64
16	4	4	538	RDMODES	RANK	128
16	4	4	506	RDMODES	RANK	256
16	4	4	613	RDMODES	RANK	512
16	4	4	594	RDMODES	RR	64
16	4	4	546	RDMODES	RR	128
16	4	4	470	RDMODES	RR	256
16	4	4	602	RDMODES	RR	512

Table 2: RDMODES option influence (to 200Hz)

Runs were performed by increasing the number of MPI processes per node from 1 to 8 as shown by figure 18 as in section 3.2.2. The same observation one can make is that for a number of nodes equal to 2, it is apparent that above 4 MPI processes per node, performance decreases, because the memory bandwidth available per node becomes saturated. Hence, the optimum number of MPI processes per node in these cases is 4. For single node runs, 8 cores per node provides the best performance.

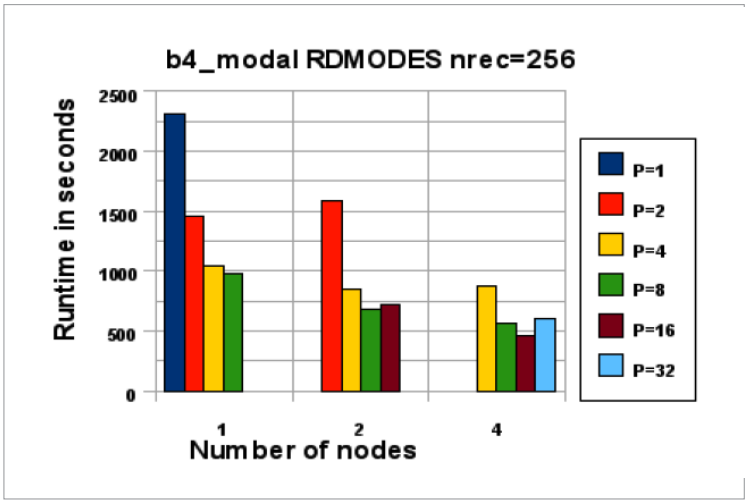

Figure 18: Benchmark runtimes by changing # of MPI processes per node (to 200Hz)

Figure 19 shows the comparison between HDMODES and RDMODES against node utilization.

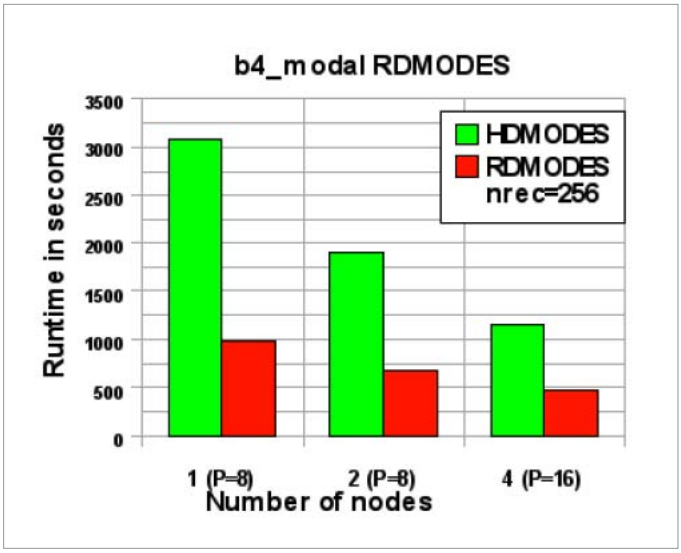

Figure 19: Benchmark results against node utilization (to 200Hz)

From the figure it appears that RDMODES is approximately 3 times more efficient.

3.2.4 Effect of frequency range to RDMODES performance

Increasing the frequency range from 200Hz to 400Hz benefits RDMODES. As in section 3.2.3, runs were performed again by increasing the number of MPI processes per node from 1 to 8 (figure 20). The same pattern observed again is that for a number of nodes equal to 4, above 4 MPI processes per node, performance decreases and the optimum number of MPI processes per node in these cases is 4.

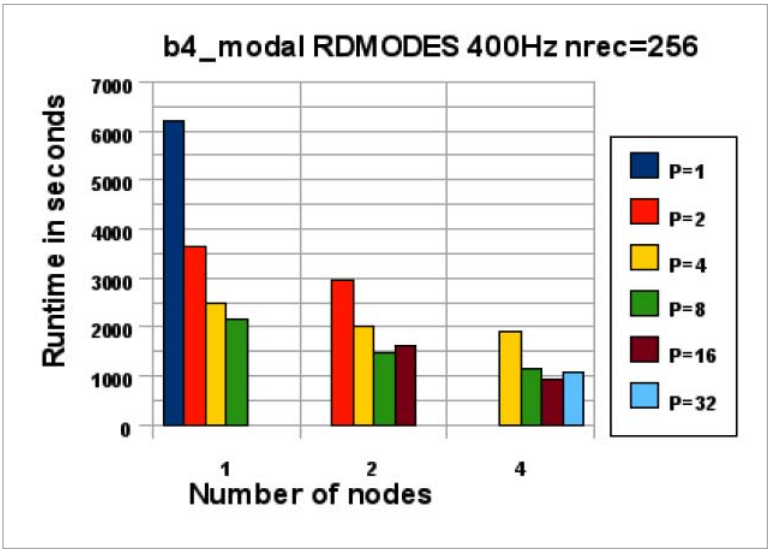

Figure 20: Benchmark runtimes by changing # of MPI processes per node (to 400Hz)

Figure 21 shows the performance of RDMODES for frequency range of 200Hz and 400Hz against node utilization. It can be seen that the larger the frequency range the better the scaling is. On 4 nodes with 16 processes the 400Hz range task is only about twice slower than the 200Hz task, despite the fact that the computational work is quadratically increased with the widening of the frequency range. Specifically the number of modes found up to 200Hz was 1019, while the number of modes below 400Hz was 3090.

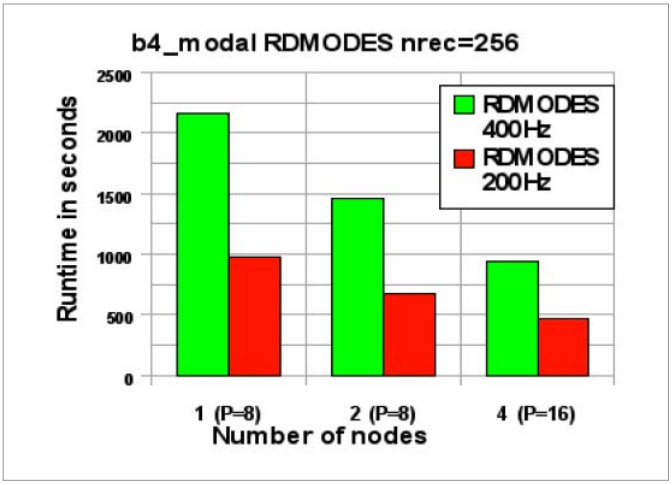

Figure 21: Benchmark results against node utilization

3.2.5 Effect of File System Distribution

Table 3 shows that if Direct Attached Storage is not available, two Network Attached Storage file systems can be used simultaneously. Using libFFIO improved performance using the same file systems. If Direct Attached Storage is available, then, one file system performs as well as two Network Attached Storage file systems of previous runs.

P	N	c	times	Description	Filesystems
8	1	8	4375	8 geometry domains	2 NAS filessystems
8	1	8	3077	8 geometry domains	2 NAS filesystems (libFFIO)
8	1	8	3171	8 geometry domains	DAS on head node
8	1	8	3150	8 geometry domains	DAS on head node (libFFIO)

Table 3: File system influence (to 200Hz)

3.2.6 Effect of Memory

Three different pairs of runs using RAM capacities of 32GB and 16GB show in Table 4 that reducing RAM on this benchmark can result in double digit runtime percentage differences Table 5.

P	N	c	times	Description	RAM
4	2	2	2612	4 geometry domains, 2 per node	32GB
4	2	2	2909	4 geometry domains, 2 per node	16GB
8	2	4	2053	frequency 2-way split, geometry 4-way striped across nodes	32GB
8	2	4	2285	frequency 2-way split, geometry 4-way striped across nodes	16GB
8	2	4	1912	frequency 2-way striped, geometry 4-way split across nodes	32GB
8	2	4	2017	frequency 2-way striped, geometry 4-way split across nodes	16GB

Table 4: Memory influence (to 200Hz)

32GB	16GB	Diff	Pct
2612	2909	296	11%
2053	2285	232	11%
1912	2017	104	5%

Table 5: Effect of memory size (to 200Hz)

The conclusion is that the more memory, the better the performance.

3.2.7 Interconnect Influence

Two different mappings of MPI processes on node schemes were run using GigE and Infiniband interconnect fabrics as shown in table 6.

P	N	c	times	Description	Interconnect
16	4	4	1220	frequency 4-way split, geometry 4-way striped across nodes	Infiniband
16	4	4	1305	frequency 4-way split, geometry 4-way striped across nodes	GigE
16	4	4	1160	frequency 4-way striped, geometry 4-way split across nodes	Infiniband
16	4	4	1205	frequency 4-way striped, geometry 4-way split across nodes	GigE

Table 6: Interconnect influence (to 200Hz)

Infiniband is faster than GigE by a single digit percentage in both runs as shown in Table 7. The optimum configuration is to use Infiniband and stripe the geometry across the nodes.

Infiniband	GigE	Diff	Pct
1220	1305	85	7%
1160	1205	44	4%

Table 7: Effect of type of interconnect (to 200Hz)

3.3 Benchmark Results on SGI Octane III

3.3.1 HDMODES Results

Runs were performed by increasing the number of MPI processes per node from 1 to 8 as shown by Figure 22. The maximum number of processes runnable per node is 8 which is the number of cores on each node's pair of Xeon quad core processors. Reading Figure 22, a run with P=32 in a column indexed by a number of nodes equal to 4 means 8 processes were run on each node, that is utilizing all 8 cores of each node. In another case, a run with P=2 in a column indexed by a number of nodes equal to 2 means only 1 process was run on each of the 2 nodes used for that job. Above 4 MPI processes per node, performance decreases, because the memory bandwidth available per node becomes saturated. Hence, the optimum number of MPI processes per node in these cases is 4.

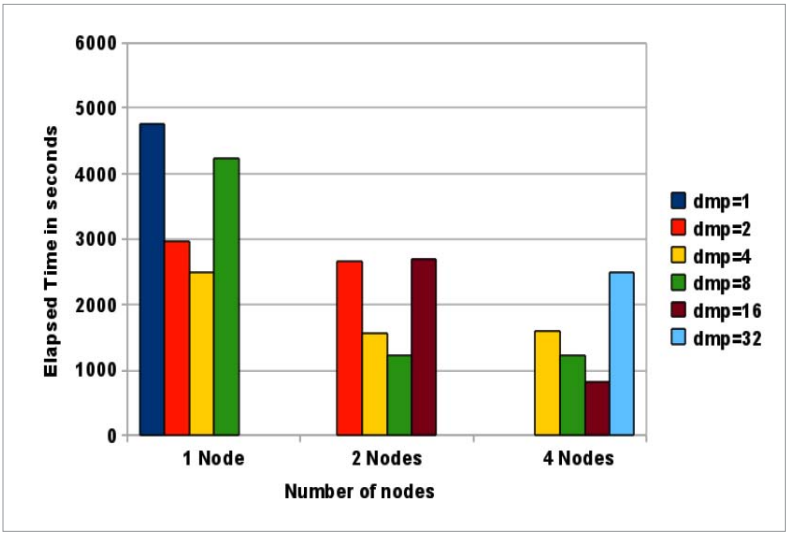

Figure 22: Benchmark runtimes by changing # of MPI processes per node (to 200Hz)

3.3.2 RDMODES Results

Runs were performed by increasing the number of MPI processes per node from 1 to 8 as shown by Figure 23 as in section 3.3.1. Above 4 MPI processes per node, performance decreases, because the memory bandwidth available per node becomes saturated. Hence, the optimum number of MPI processes per node in these cases is 4.

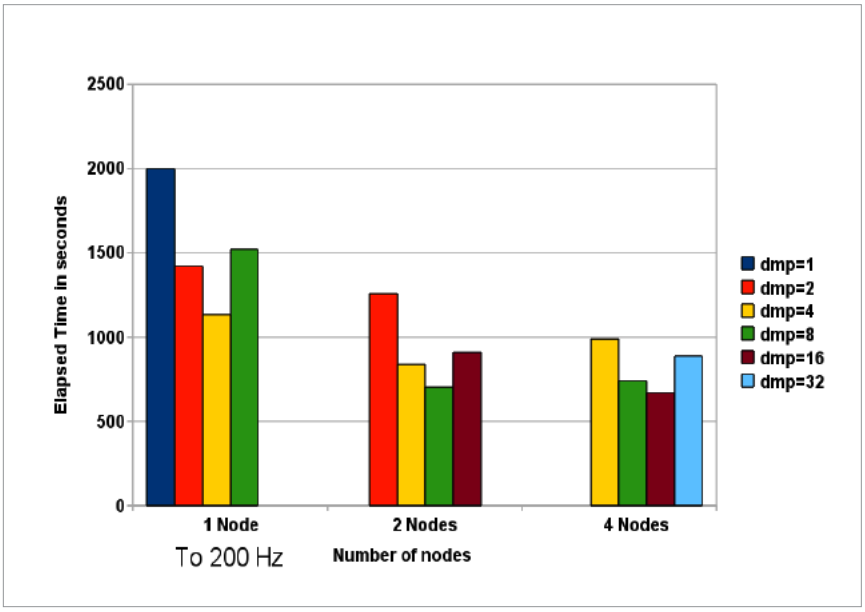

Figure 23: Benchmark runtimes by changing # of MPI processes per node (to 400Hz)

3.3.3 RDMODES Results 400Hz

Increasing the frequency range from 200Hz to 400Hz benefits RDMODES. As in section 3.2.3, runs were performed again by increasing the number of MPI processes per node from 1 to 8 (Figure 24). Above 4 MPI processes per node, performance decreases and the optimum number of MPI processes per node in these cases is 4.

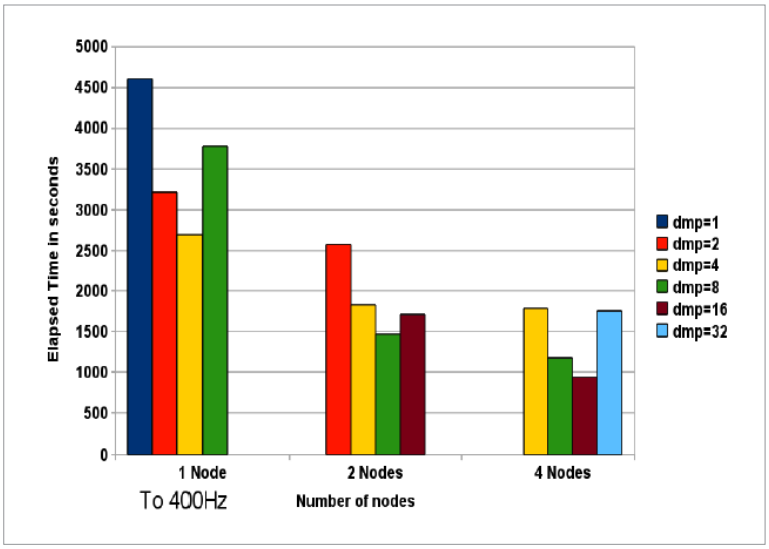

Figure 24: Benchmark runtimes by changing # of MPI processes per node (to 400Hz)

3.4 Benchmark Results on SGI UV100

3.4.1 RDMODES Results DMP+SMP

Runs were performed by increasing the number of OpenMP threads per MPI process from 1 to 4 as shown by Figure 25. Performance increases for 2 OpenMP threads per MPI process up to 8 MPI processes where 2 OpenMP threads slow down performance.

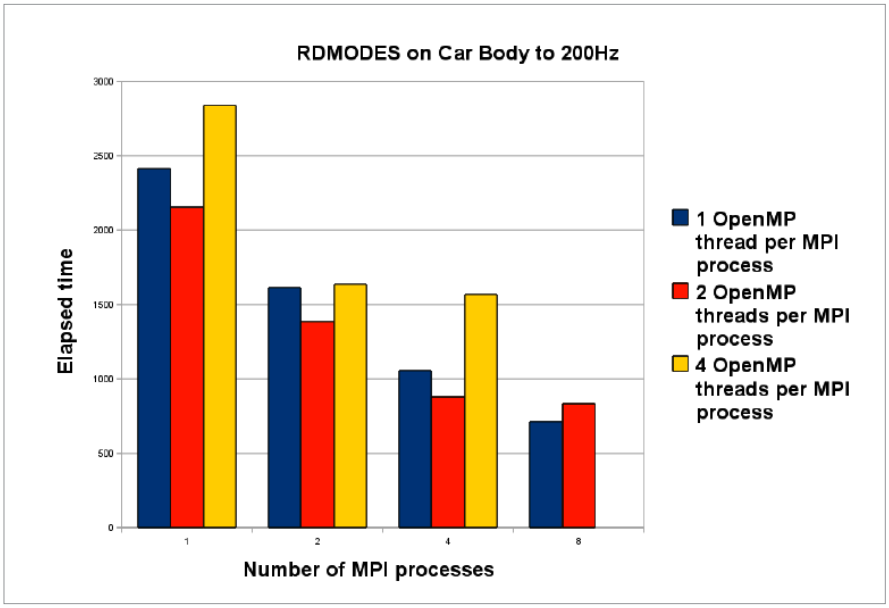

Figure 25: Benchmark runtimes by changing # of MPI processes per node (to 200Hz)

Conclusions

This study showed that memory per core, high speed interconnect, memory speed and placement of data file on a high performance file system strongly influence analysis performance. At the same time, by using the right combination of NX NASTRAN parallelism features, including RMODES, most efficient use of either shared or distributed memory systems can be achieved to optimize throughput or turnaround times on both Direct Attached and appropriate Network Attached Storage.

These insights and optimizations can be used to architect a solution using the full range of systems from SGI to meet complex analysis requirements. NX NASTRAN offers integrated distributed parallel computational solutions applicable to most advanced computer architectures. SGI offers advanced hardware platforms to meet complex industrial analysis requirements. Using the right combination of NX NASTRAN features and SGI advanced hardware architectures resulted in excellent analysis turnaround times in automobile industry benchmark.

Attributions

References

[1] Olivier Schreiber, Scott Shaw, and Louis Komzsik. “NX NASTRAN on Advanced SGI Architectures”. In Proceedings of Siemens PLM Connection Americas User Conference, Dallas, TX, June 2010.

[2] Louis Komzsik. “Industrial FEA, Evolution and Current Challenges”. In Proceedings of NAFEMS World Congress, Crete, May 2009.

[3] Louis Komzsik. What Every Engineer Should Know about Computational Techniques of Finite Element Analysis, Second Edition. Taylor and Francis, 2009.

[4] SGI. SGI Developer’s Guide. Silicon Graphics International, Fremont, California, 2009.

Corporate Office
46600 Landing Parkway
Fremont, CA 94538
tel 510.933.8300
fax 408.321.0293
www.sgi.com

North America +1 800.800.7441
Latin America +55 11.5185.2860
Europe +44 118.912.7500
Asia Pacific +61 2.9448.1463