sgi®
INNOVATION
FOR RESULTS™

White Paper

Solving Large Problems on Large Computer Systems

Hardware Considerations for Engineering Productivity Gains with Multi-Discipline Nastran Simulations

Table of Contents

Abstract

High fidelity Finite Element models increase demands on computer hardware. Different numerical algorithms required for multi-discipline analyses increase the complexity of system choice. We examine the resource usage and performance of solvers in the context of Multi-Discipline Nastran on SGI Altix® IPF and XPF systems. We report issues that can arise in solving industry strength problems. We discuss the merits of different computer architectures for explicit dynamics, NVH, linear and non-linear static analysis methods as well as file system performance.

1.0 Introduction

Simulations of complex engineering systems such as aerospace or automotive vehicles require modeling the important interactions between the physical phenomena and the vehicle. These interactions make the vehicle a synergistic whole. Taking advantage of that synergy is the mark of good design. However the web of interactions is difficult to untangle effectively in the time constraints of product development. Multi-Discipline (MD) Nastran is intended for the synthesis of complex engineering systems.

MD Nastran R2 release introduces powerful new multidiscipline simulation features and enhancements to address challenges in the areas of:

- Automatic sub-structuring methods ACMS for NVH
- Interior and exterior acoustics incorporating MSC Actran infinite elements
- Impact, crashworthiness and occupant safety with explicit finite element analysis incorporating LS-DYNA's Largrangian and MSC Dytran's Eulerian forumaltions
- Contact and nonlinear analysis using MSC Marc
- Aeroelasticity and rotordynamics with MSC Nastran
- Process simplifications to speed modeling work flow
- Multidiscipline design optimization

With the increasing model sizes generated by engineering teams, performance of MD Nastran for very large scale NVH and impact models continues to be critical. Total system superiority is mandatory to obtain efficient solutions. Achieving scalable parallel processing (DMP, SMP), true 64-bit addressing and floating point performance, and solver robustness is imperative. In this context, it is important to emphasize that application productivity is not tied to parallel capability or other standard compute metrics but is also closely related to file system.

The last decade saw the commoditization of both the computer hardware and the operating system software. According to the Dresden IDC report dated June 2007, 66% of today's high performance servers are running a variant of Linux, followed by proprietary UNIX software and Windows. On top of the commercial Linux distributions such as RedHat and SuSe, some system vendors provide additional features. SGI supports the XFS (Reference 1) journal file system, DLPACE (Reference 2), and FFIO (Reference 3) as well as other useful management features in the Propack series of releases for Linux.

Server architecture is currently dominated by the x86-64 processor families (63%) which include both Intel Xeon 5100 and AMD Opteron series. The remaining servers are RISC or EPCI (Intel Itanium2) processor based. The x86-64 based systems are either dual-core or quad-core processors. The multi-core processors may not necessarily provide the right balance between CPU processing and memory bandwidth for running specific high performance engineering applications. As a result, system vendors, such as SGI, offer both 'density' and 'bandwidth', and 'super bandwidth' configurations to meet customer solution requirements.

In this paper, we compare MD Nastran performance on an Itanium2 A450 and a cluster of Xeon 5160 based SGI Altix® XE's. These are two very different architectures – VLIW vs superscalar processors and ccNUMA SSI vs cluster. On paper, the Xeon have much higher clock speed, twice bandwidth per memory controller, and significantly better SPEC 2006 integer performance. Yet the A450 offers a large amount of shared memory and matured, well tuned, MD Nastran application software. The storage options on A450 and XE also are very different to make the comparison interesting.

2.0 Size Matters

Small models, can fit better in the memory on the processor chip (cache), and therefore benefit directly from an increase in processor speed, regardless of the solver used. With a smaller number of finite elements, the interior floating point dominated processing loops are short and the integer requirements proportionally balanced. This combination fits the Intel Xeon processor architecture. This is demonstrated by the performance illustrated in Figure 2-1. These examples of small models run with several different MSC Software Nastran and Marc solvers run faster on the SGI Altix® XE Intel Xeon based system.

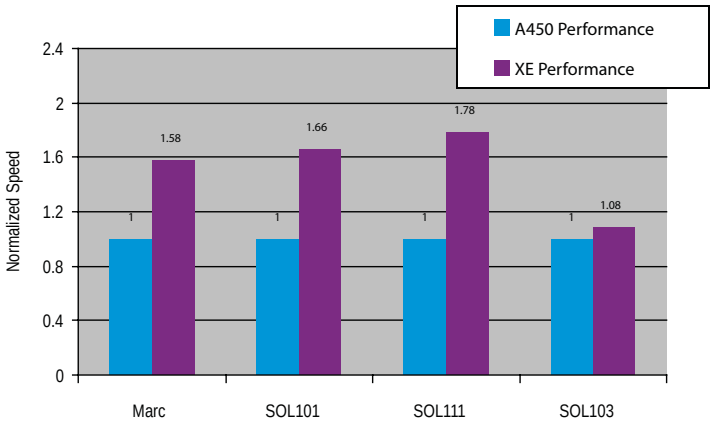

Figure 2-1. Small Model Comparisons

The Marc model in the chart has about 12000 elements and nodes. The SOL101 job consists of only shell elements and is restricted to 989K degrees of freedom. SOL111 is a 2 Million degree of freedom shell element model that computes 1000 nodes and 1000 frequency points. SOL103, which uses the same model as the SOL101 example, only solves for 15 eigenmodes.

As the model size increases beyond the limits of effective caching, more complex system operations including the memory controller, RAM, disc, and interconnections, the quality of the compiler, application specific source code tuning, and solution algorithms would play a more prominent role in the job performance. Comparisons in the following sections will consider these factors using the A450 Intel EPIC processor based systems and the SGI Altix® XE Intel Xeon processor based architecture.

Due to internal book-keeping, Nastran and Marc with 32bit integers (i4) are limited to 8GB memory per domain. Large problems encourage the use of 64 bit integer (i8) versions of Nastran and MARC for larger addressable memory. Today's industry strength models and systems need the i8 codes to run efficiently. However, it is determined in this and previous work that some of the solution sequences do not run as efficiently with the i8 code compared with the i4 counterpart due to compiler maturity. For these incidences, MSC has identified and fixed i4 integer overflow problems in some 'hot spot' areas, which are mainly related to global data structure manipulations, such as matrix re-ordering and domain decomposition. The METIS re-ordering code in Nastran also has a path to allow i4 code to temporarily use more than 8GB memory for the duration of the METIS execution. This is typically insignificant in comparison to the overall run time.

Marc has a similar issue with the i8 codes, but due to the resource usage scale-down from domain decomposition that is often used by Marc, the problem is not as significant.

3.0 Linear Equation Solvers

The solution of a linear system of equations, for example: [A]*{X} = {B}, is one of the basic tasks in FEA. In most cases [A] is the global stiffness array, {B} is either a force vector or a block of force vectors and {X} is the solution corresponding to {B}. The choices of linear equation solvers in MD v2007R2 Nastran include the conventional MSC LDLT sparse direct solver, the CASI iterative solver, a new TAUCS symmetric and a new UFMPACK asymmetric sparse direct solver. The sparse direct solvers are basically stable due to numeric pivoting in the code, however the iterative solver has its limitations, per Nastran release guide (Reference 4).

For optimal performance, a sparse direct solver is implemented with a matrix multiplier like a BLAS3 type math kernels, whereas an iterative solver is typically done with a matrix-vector multiplier like BLAS2 type kernels. Matrix multiplier kernels are stable and easily blocked for better data reuse and thus better performance. However, matrix-vector kernels are memory and bandwidth intensive and resemble a bcopy/memcpy code.

To illustrate the pros and cons of the differing linear equation solvers on the two hardware architectures, we ran three SOL 101 jobs. The first job is an 1.92 mil DOF engine block of only solid elements. The second job is also solid element dominant but has 3.33M DOFs and roughly 10 percent of the elements are shells. The third job is a trimmed car body that is made up mostly of shell elements and 1.83M DOFs. Multiple force vectors (i.e. Nastran load sub-cases) are also tested to demonstrate how the CASI solver time increases with the number of loads, since an iterative solver would handle only one right hand side vector at a time.

The results of these three tests are shown in Figures 3-1, 3-2, and 3-3. Note that the CASI solver fails to converge for the 3rd test, and the iterative solver is most effective with the pure solid element job. The i4 and i8 codes are roughly equal in speed for the Nastran SOL101 linear static analysis, and the shell element job with 1.83M unknowns is still not large enough to make A450 run faster, even though the A450 has faster BLAS3-like kernels for the sparse direct solvers.

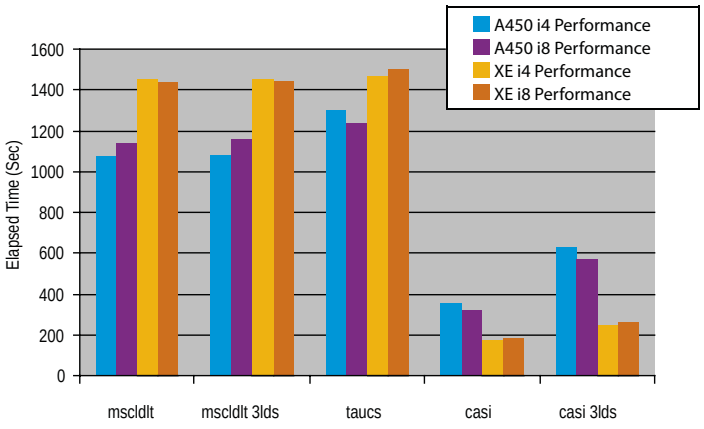

Figure 3-1. SOL101 of Solid Elements

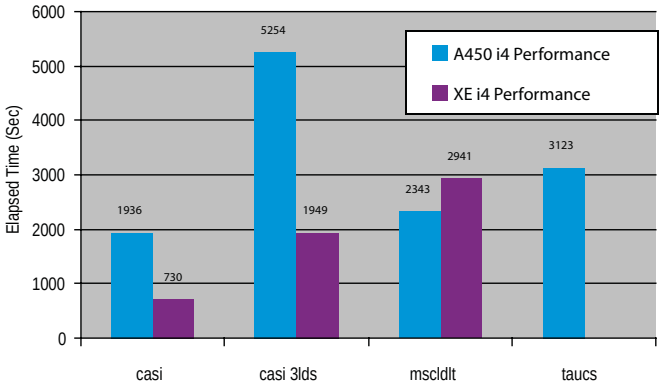

Figure 3-2. SOL101 of Mixed Elements

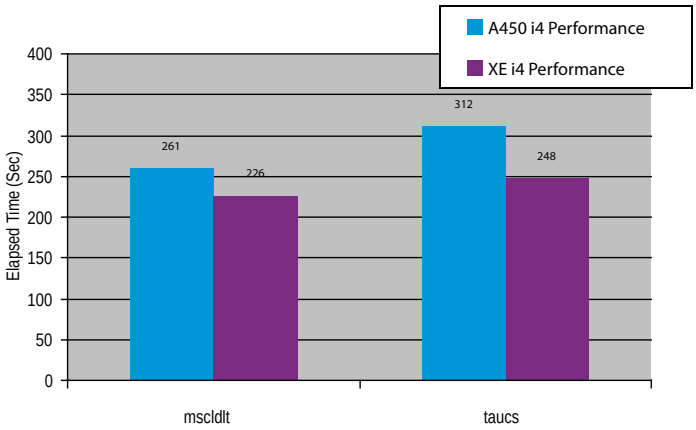

Figure 3-3. SOL101 of Shell Elements

4.0 Non-linear Equation Solvers

A non-linear problem arises when the global stiffness matrix no longer represents a linear relation between the force and the displacement vectors. In this instance, a Newton-Raphson method or a method akin to it is necessary to bring the imbalanced force down to equilibrium. The numerical procedure is similar to that of linear static problems, except that the solution of the system of equations, [A] * {X} = {B}, is repeated as many times as is required to reach equilibrium. Moreover, a non-linear solution would be capable of dealing with only one load vector at a time.

MSC.Marc has solvers 2, 6, 8, and 9. We test all the solvers against a bio-engineering model that has very high floating point content, (i.e. about 700 Gflops per solver pass) for up to 16 cores on A450 and XE. To use solver 6 with domain decomposition, the grid points in the model have to be numbered sequentially.

If there are contact bodies in the model, the element IDs need to be numbered contiguously as well. These requirements can generally be met by reading the non-contiguously numbered model into MSC Mentat first, then hitting the 'RENUMBER ALL' button and writing out the new input file, followed by minor tweaking. Note that solvers 2 and 8 are not subject to these restrictions and solver 9 is subject to restrictions for the CASI iterative solver.

Our test results are shown in Figures 4-1, 4-2, and 4-3. The solver 9 results are not shown since they are too far behind those of the other solvers. The solver 2 fails to converge on 1 core, thus, that case is not shown.

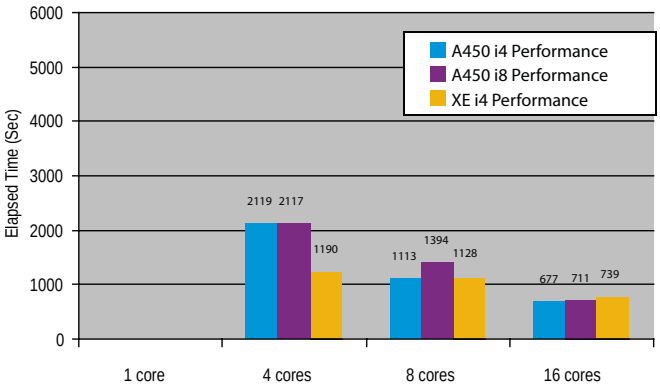

Figure 4-1. Marc Solver2 Results

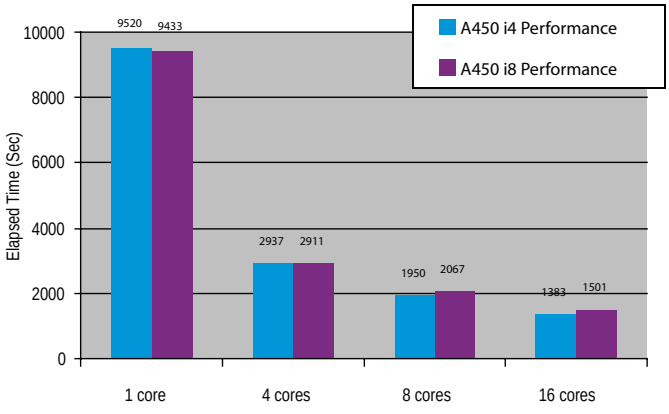

Figure 4-2 Marc Solver6 Results

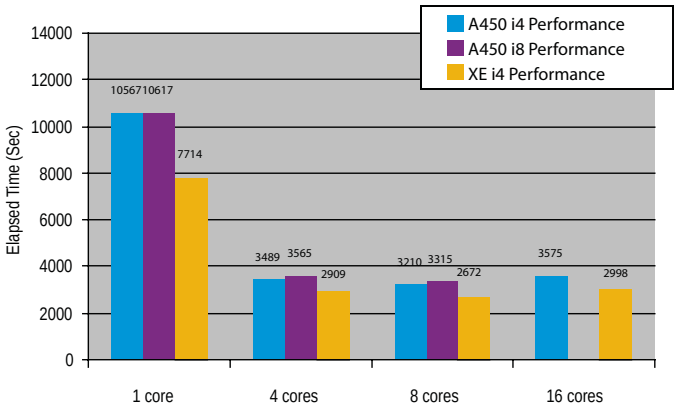

Figure 4-3. Marc Solver8 Results

5.0 Lanczos Eigenvalue Solver

The Lanczos eignevalue solution in Nastran, which traces its roots to the BCSLIB-EXT Lanczos solver, has been widely implemented to perform vibration and NVH analyses for all types of structures. The Lanczos method operates on blocks of long and thin matrices that resemble vectors. Due to this, and the limited memory of Nastran HI-Core, the Lanczos solution is typically memory bandwidth and IO intensive.

Due to the unique shape of the Lanczos blocks, writing efficient BLAS kernels for Lanczos block operations can be challenging. To date, the A450, or rather, the Itanium 2 processor, is more adept for running Nastran Lanczos analysis jobs due to better-optimized BLAS kernels.

Figures 5-1 and 5-2 show the performance of the a.m. 3.33M DOF trimmed body model for both NVH (to 250HZ) and vibration (to 20HZ) tests on one core. For these Lanczos jobs, A450 is significantly faster than XE, however, the i8 code also slows significantly for the NVH case. Further examples of the sluggishness of the i8 can be seen in Table 5-1, which shows the timing results of a 34M DOF car body for about 2000 modes that is run on an A450 cpuset with 16GB physical memory and a 24 SAS disks file system.

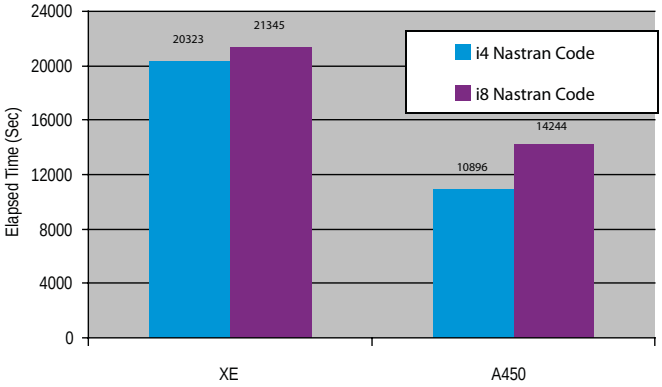

Figure 5-1. 250HZ Lanczos Analysis

Nastran	Elapsed Sec	CPU Sec	IO Wait	Utilization %
i4 code	180125	160002	20123	88
I8 code	235306	218771	16535	92

Table 5-1. Very Large Lanczos Analysis

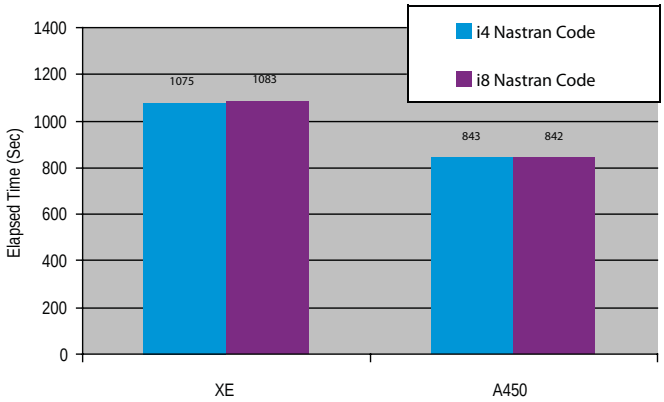

Figure 5-2 20HZ Lanczos Analysis

6.0 Explicit Dynamics Solvers

MD Nastran SOL 700 is a general purpose, transient dynamic, non-linear, explicit (with implicit capability), finite element analysis software based on LS-DYNA and MSC Dytran solvers. John Hallquist originally wrote DYNA-3D for Lawrence Livermore National Laboratory, which was subsequently released to the public domain. MD Nastran SOL 700 features include:

- Highly nonlinear:
  - Changing boundary conditions with time such as contacts between parts
  - Large deformations
  - Nonlinear (non elastic) materials
- Transient dynamics
- Important inertial forces
- Finite element analysis
- Explicit time integration

SOL 700 is used in many industries:
- Automotive
- Aerospace
- Manufacturing and
- Bioengineering
- Consumer

and disciplines:
- Crash, impact
- Metal forming
- Blade containment
- Bird strikes
- Drop testing
- Plastic, glass forming

Explicit Nonlinear Dynamics was introduced in MD R1 Nastran as an executive control statement SOL700. It activates an explicit nonlinear transient analysis integration scheme using dytran-lsdyna. It may also be used for implicit static analyses using MD Nastran nonlinear implicit solver such as SOL 400 or SOL 700 implicit solver based on LS-DYNA double precision version. In MD R2 Nastran, the nastran input undergoes a filtering process for model accuracy and then is directly mapped into LS-DYNA memory by means of a structured neutral file. The SOL 700 implicit and explicit solvers will streamline complex, multi-discipline sequential simulations such as pre-stress analysis (implicit to explicit), springback effects (explicit to implicit) or other simulations that will require switching between solvers.

SOL700 is intended for engineers and analysts who have constructed an MSC Nastran finite element model for a purpose other than impact. This avoids having to read the MSC Nastran model into a GUI, translate it to LS-DYNA or MSC Dytran, and thus risk losing or not properly translating some MSC Nastran input data. Once one has completed the explicit simulation, standard LS-DYNA results files such as d3plot as well as standard MD Nastran files are available for post processing.

SOL700 has three ways of solving static problems:

- Dynamic relaxation: The input is applied as a step function and large damping is added. The solution is run until approximate steady-state values are obtained. (classic method)
- Slow buildup: The static load is ramped slowly from zero to full value over a period of time long enough that no important natural frequencies are excited. No extra damping is added. (for exact results)
- Slow buildup with extra damping: This method is like the previous method except that some extra damping is added; thus, the final run time can often be reduced.

Unless explicitly specified, SOL 700 will be executed from MD Nastran on any computer system where it is installed, licensed, and accessible from the directory where the MD Nastran input data resides as prescribed on the first line in a file named sol700.pth. Using this file, MD Nastran will create a command line to start the SOL 700 explicit solver.

For customers comparison, standard benchmarks have been developed over the last twenty years and are available on http://www.topcrunch.org. Figures 6.1 and 6.2 illustrate the models discussed in this paper.

Figure 6.1. Neon refined revised

Figure 6. 2. 3 Vehicle Collision

With the advent of multi-core processors, new rules were developed and as of May 7, 2007, all cores for each processor must be fully utilized. Benchmarks of multi-core processors using only a core subset per processor will no longer be posted. The objective is to reduce clutter on the site so that end-users will not have to sort through every permutation of core usage for a fixed number of cores, and try to understand which cores are idle, and which are not.

SOL 700 sequentially goes through the following phases:

1. Initialization:
   1.1 reading input file[s],
   1.2 allocating memory
   1.3 initializing variables
   1.4 domain decomposition
2. Element-processing
3. Contacts
4. Rigid bodies

The explicit time integration computation kernels used in SOL 700 involve less round-off errors than implicit solver computations. Consequently, either Double or Single Precision may be used for short simulations where these round-off errors do not accumulate over many time steps. Double Precision is necessary for models with more than 5 million elements which cannot be resolved by single precision floating point numbers.

Resource requirements are dependent on problem size and complexity of the physics simulated:

- RAM: Total usage is proportional to the square of 2D shell elements (SP GB=1*E^2, DP GB=4*E^2) resulting in the 2GB limit of 32 bit OS memory addressability reached with 1M elements in Single Precision and 700k elements in Double Precision.
- RAM per process decreases as the number of processes increases potentially leading to super-linear speedup opportunities as the computational domain matches the size of the on-chip memory.
- IO storage: dependent on user options on SOL 700 output files and restart specifications.
- Communication bandwidth: SOL 700 DMP (Distributed Memory Parallel) has small and decreasing message sizes as the number of processes increases and is communication latency limited.

Platforms attributes have complex effects on SOL 700 benchmarks: SOL 700 and SOL 700 DMP are respectively OpenMP and MPI parallel capable and, as other Computational Structural Mechanics explicit CAE applications, are very sensitive to SMP and DMP technologies and their assorted interconnects.

Figure 6.3 and Figure 6.4 show for both neon refined revised and 3 Vehicle collision how for high process counts IA64 processors with NUMAlink outperform x86_64 with Infiniband.

Figure 6.5 shows how GigE under-performs Infiniband on the neon_refined_revised benchmark on the same x86_64 processors.

Double Precision processing involves more registers, cache space and memory bandwidth than Single Precision processing. Therefore the influence of processor, processor Chipset, FSB in addition to DMP and interconnects is greater.

Figure 6.6 shows for x86_64 processor ratios of Double Precision to Single Precision elapsed times as high as 1.95 for the neon refined revised benchmark whereas Figure 6.7 for IA64 processor shows ratios as low as 1.16 even in the case of the larger 3 Vehicle collision benchmark. The ratios also get lower as the number of processes increases.

Process placement on cores within processors affects Front Side Bus, cache and socket use. For example, the Woodcrest processor/socket has 2 cores sharing 4Mb of cache. The Clovertown socket is not native quad-core but has 2 Woodcrest processors combined on the same socket for a total of 4 Cores, 2 x 4Mb L2 caches and 1 FSB. 2 processes placed on the same processor may therefore share or not share a single L2

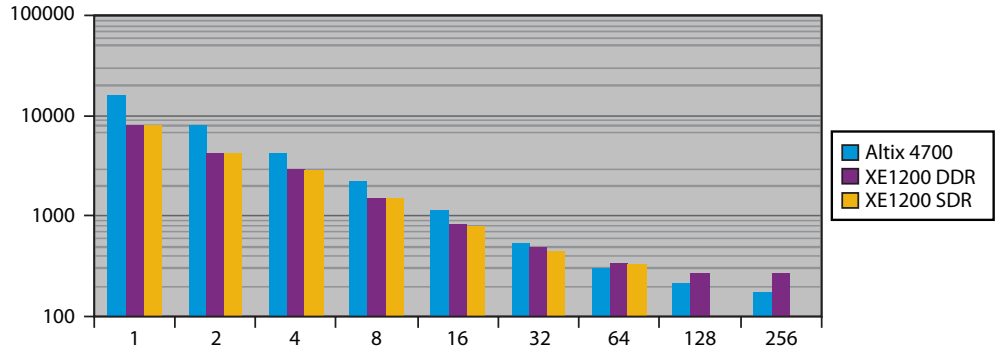

Figure 6.3. Neon refined revised elapsed times vs process count

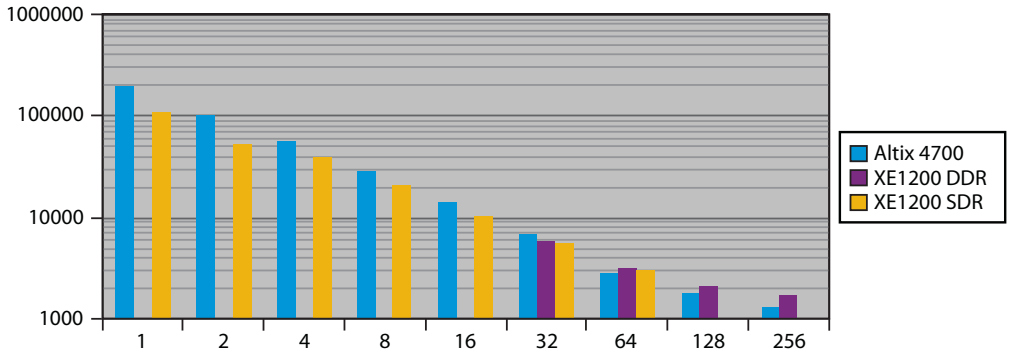

Figure 6.4. 3 Vehicle Collision elapsed times vs process count

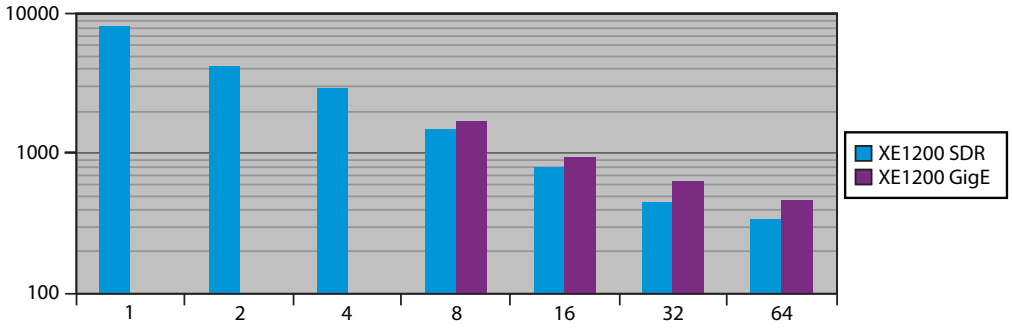

Figure 6.5. Neon refined revised elapsed times vs process count

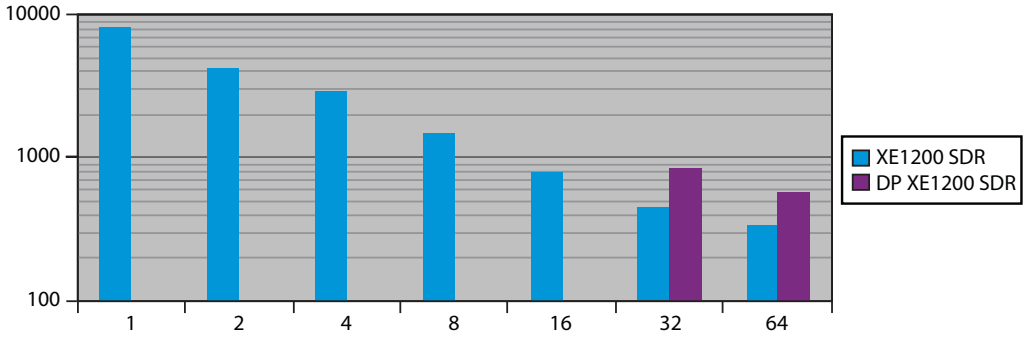

Figure 6.6. Neon refined revised elapsed times vs process count

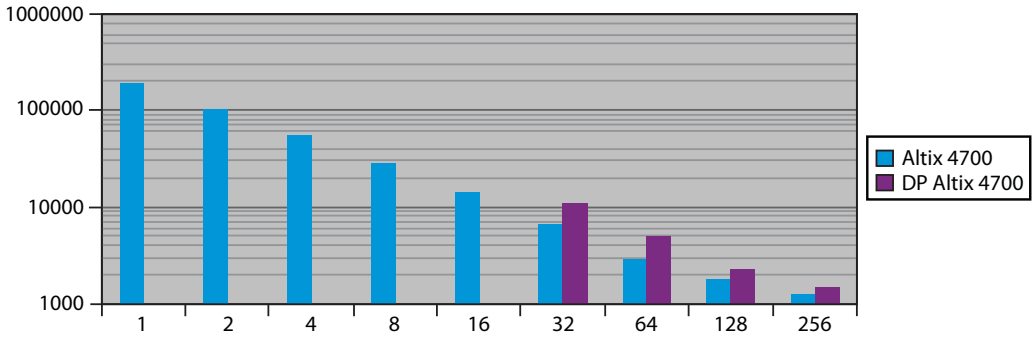

Figure 6.7. 3 Vehicle Collision elapsed times vs process count

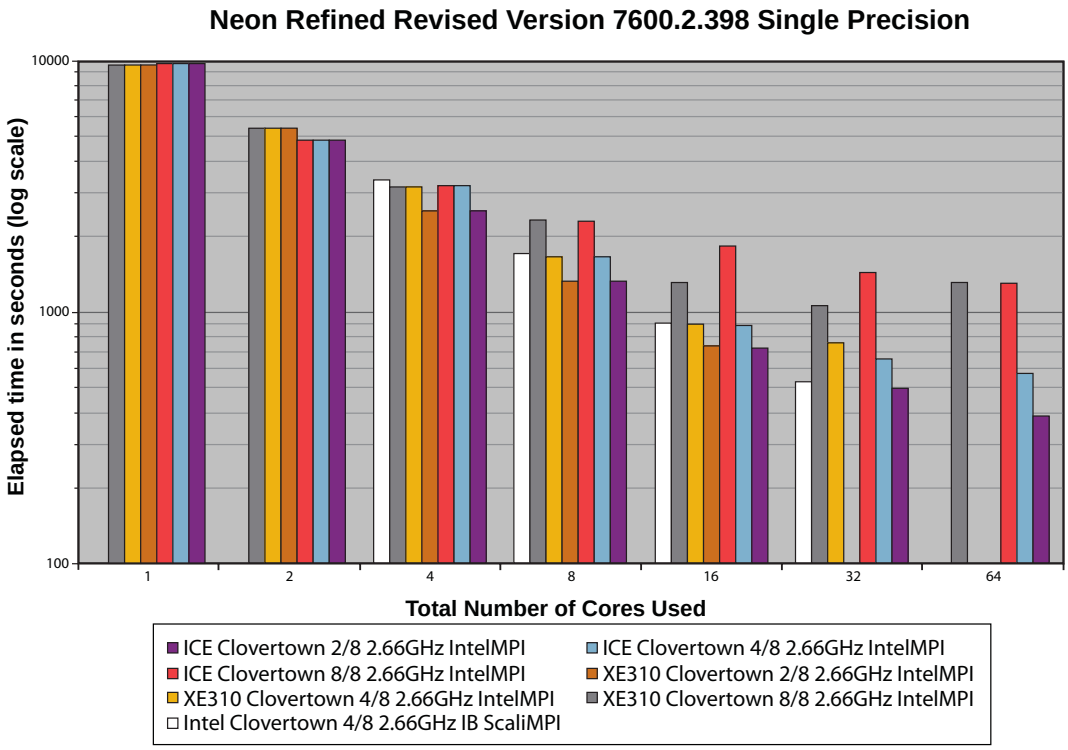

Figure 6.8

cache and 4 of them on a dual processor may or may not share the same FSB. With appropriate process pinning, for cache sensitive applications 4 threads per node on Clovertown may be better than 4 threads per node on Woodcrest system offsetting frequency difference.

Figure 6.8 shows for the neon refined revised benchmark the trade offs which can be made between core utilization and performance in the case of the Clovertown processor for three different systems. The ratios in the legend document how many cores out of 8 are used per node. Performance is evidently higher with more nodes and less cores per node utilized to run a job with a given number of processes.

7.0 ACMS Solver

MSC has implemented ACMS (Reference 5) as a complement to the Lanczos eigensolver for modal analysis. This is the latest product in a long evolution from the Subspace eigenvalue solver. The ACMS solver uses a component mode synthesis technique to approximate the global eigenvalues and modes, and is able to achieve efficient resource usage and better performance over the Lanczos method, with acceptable numerical accuracy.

In addition to ACMS, for frequency response calculation in modal analysis, MSC also has implemented a FASTFR (Reference 6) method as a complement to the sparse direct solver. In Figure 7-1, we compare ACMS performance with both the fastfr, which works best with a low structural damping and a large frequency range, and the direct methods. The model that is tested has 6.3 Million degrees of freedom, 3600 eigenmodes, 130 load sub-cases, and 700 frequency points. Both ACMS and FASTFR use BLAS3 style matrix multiply kernels extensively.

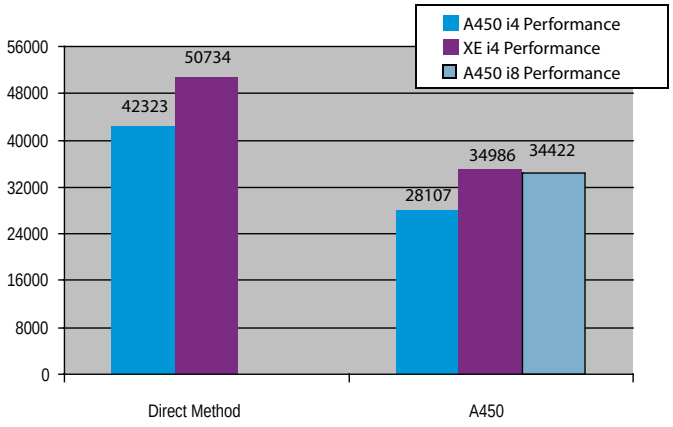

Figure 7-1. SOL111 NVH Analysis

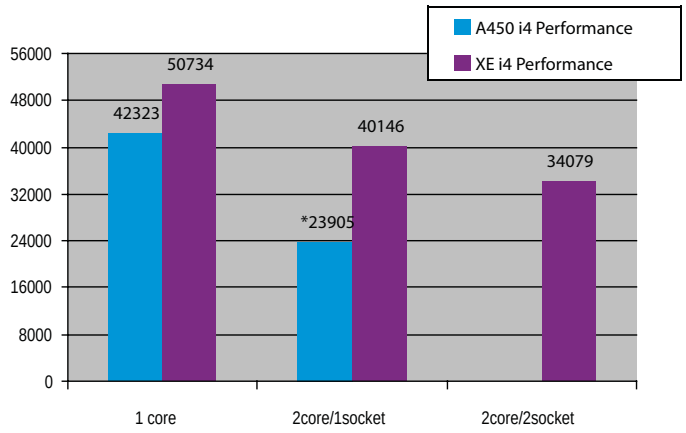

Figure 7-2. SOL111 NVH 2 Core Scalability

The results in Figure 7-1 represent the 1 core execution times. The 2 core performance is compared in Figure 7-2. We have run the 2 core job on a XE240 twice and placed the 2 threads on either 1 or 2 sockets to show the Xeon cache sensitivity. The A450 job is run on only 1 socket. Only the direct method and the i4 codes are used. As a reference, we have left the 1 core performance in the chart.

*This 2 core test is done on a 1.67ghz dual core Montvale processor.

8.0 Multi-Discipline (MD) Considerations

The synergistic integration of linear and nonlinear analysis, implicit and explicit methods, design optimization methods and high performance computing in MD Nastran provides for superior value in multiple ways including:

- Ability to chain various analysis types in a single job through multiple sub-cases in case control – first sub-case could be nonlinear static followed by a second sub-case with nonlinear transient – the initial condition from a previous static analysis for the current nonlinear transient analysis step;
- Efficient transition from linear to nonlinear solutions exploiting the contact definitions that are universal across solution sequences; This is critical to address the constraint of a linear solution such as NVH (SOL 103 and SOL 111) cannot have nonlinear contact modeling that is used with a nonlinear explicit solution (SOL 700);
- Improved ability to simulate complex interactions in product designs (such as between NVH and crashworthiness in a vehicle system design); and,
- Reducing the number of interdisciplinary design iterations through a tighter and more accurate coupling of disciplines that drive the design process.

The above benefits are exploited with the multidiscipline analysis and design optimization process with MD Nastran, shown in Figure 8-1, for a vehicle system that involves multiple discipline considerations including structural stresses, dynamics, acoustics (NVH), and crashworthiness.

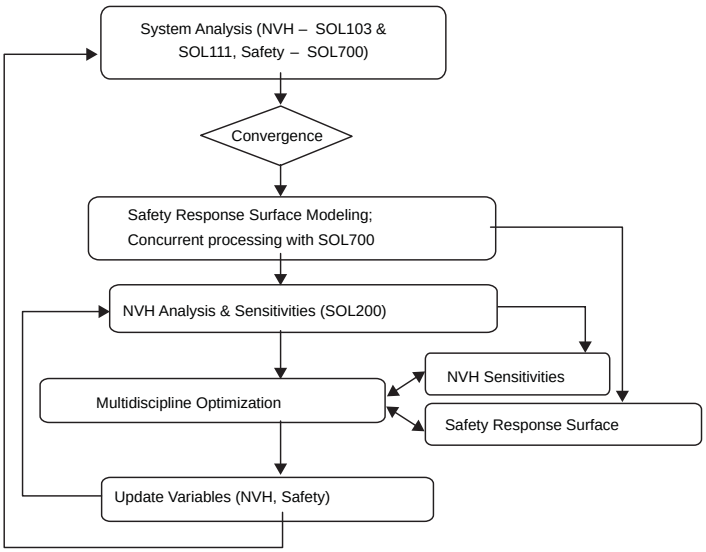

Figure 8-1: MDO Process with MD Nastran involving NVH and Safety

Until recently such complex MDO simulations where performed by chaining several point solutions in a sequential manner and using process integration and design optimization tools (Reference 7).

Clearly, the value that MD Nastran brings to the forefront is to couple various disciplines into a single simulation environment that enables more realistic and accurate simulations as well as faster solution time thus facilitating true multidiscipline analysis and design optimization. Due to the level of complexity and dimensionality associated with such MDO simulations, high performance computing with high throughput is essential to solve such problems in a realistic time so as to impact the product development cycle.

Further, the heterogeneous mix of simulations in MD Nastran exhibit a range of HPC resource demands. The implicit linear analysis solvers for dynamic NVH response requires high rates of memory and IO bandwidth with processor speed as a secondary concern while explicit dynamics solvers for impact crash simulation benefits from a combination of fast processors for the required element force calculations and a high rate of memory bandwidth necessary for efficient contact resolution. Hence the realistic need for “balanced” HPC environments to support the variety of multidiscipline simulations within MD Nastran. For today’s economics, these HPC resources such as CPU cycles, memory, system bandwidth and scalability, storage, file and data management – must deliver the highest levels of engineering productivity and HPC reliability that is possible from a platform environment.

9.0 Input and Output Economy

Large Nastran jobs read and write large files. According to some compiled FFIO exit statistics on A450, a Nastran IO stream typically has an average IO bandwidth consumption of 50 - 250 MB/s over the lifetime of the job. Therefore, for a Nastran server, storage hardware usually is a significant part of the system cost. On a large SSI machine the choice for storage hardware is obvious. That is, a locally attached file system is the logical way to go, and the file system can be shared by many jobs that are run on the SSI to amortize its cost.

On a cluster, however, it would be impractical to configure a high-bandwidth file system for each of the nodes due to the small number of jobs that will utilize the file system. The alternative would be to have a globally shared file system. To share a file system, each compute node needs to configure extra interconnect cards to sustain the desired IO bandwidth, and the storage hardware can be concentrated in an IO 'head' node, or distributed over a number of nodes. In any case, a shared file system is more expensive than the equivalent local file system due to the interconnectivity and the global file serving overhead.

In the following paragraphs, we perform a simple analysis of the local storage option for its cost effectiveness. With some experiments, we'd like to prove the following:

- A single standalone job would run 'at speed' if sustained bandwidth of the file system, say, as measured by lmdd, exceeds a certain threshold.
- Simultaneous jobs would run 'at speed' if the sum of the average Io bandwidth of each individual job in the through-put mix is significantly less than the sustained bandwidth of the file system.
- N file systems with the threshold bandwidth, where N is the number of jobs in the through-put mix, are more expensive than 1 shared file system with a large bandwidth.

In Table 9-1, we determine the 'threshold' file system configuration for a SOL106 Nastran job that has roughly 700K degrees of freedom, and consists of mostly CTETRA solid elements. This job is run on file systems of 2, 4, 8 and 16 disks sequentially, with 200Mw Nastran HI-CORE memory and a FFIO cache of 2.5GB. The 16 disk file system sustains an IO bandwidth of about 900MB/s for both read and write, and the 2 disk file system gets about 150MB/s.

According to the FFIO exit stats, the average IO bandwidth consumption of the job is only 55MB/s. However, while the job is running on the 16 disk configuration, SGI PCP, ie. Performance Co-Pilot, tool (Reference 8) shows there are a few burst periods that consume 400MB/s or 750MB/s IO bandwidth. Therefore, it is no surprise that there is almost no IO wait time for the 16 disk job, and the 8 disk job is pretty good and the 4 disk job is almost 'good enough', as far as IO wait is concerned.

For efficient throughput, we run 8 identical copies of the job on 8 different CPU nodes on the 16 disk file system first. With a dedicated node per job, there would be no memory bus contention, and the throughput performance degradation would be largely due to Linux scheduling and file system contention issues. For the second test, we run the 8 copies on 4, instead of 8, nodes. Therefore, in addition to scheduling and file system contention, the two jobs on a node need to fight for the memory bandwidth, and that result in an increase in CPU as well as IO wait times. The throughput results are shown in Tables 9-2 and 9-3. Note that we have used the DPLACE tool to ease the scheduling problem in these tests.

The average elapsed time of the 8 on 8 node throughput jobs is roughly equal to the standalone time on 4 disks. Therefore, for this particular example, a SSI with a 16 disk file system that is shared by 8 jobs is roughly equivalent to 8 separate nodes with a 4 disk file system each. In other word, the SSI needs only half the IO hardware to perform at the same level.

File System	Elapsed Sec	CPU Sec	IO Wait	Utilization %
2 disks	6028	5236	792	87
4 disks	5643	5241	402	93
8 disks	5413	5242	171	97
16 disks	5323	5255	68	99

Table 9-1. Threshold Performance

Throughput Job	Elapsed Sec	CPU Sec	Utilization %
Copy #1	5734	5326	93
Copy #2	5501	5239	95
Copy #3	5710	5260	92
Copy #4	5678	5244	92
Copy #5	5675	5263	93
Copy #6	5537	5243	95
Copy #7	5709	5258	92
Copy #8	5469	5246	96

Table 9-2. 8 on 8 Throughput Test

Throughput Job	Elapsed Sec	CPU Sec	Utilization %
Copy #1	6030	5586	93
Copy #2	6027	5583	93
Copy #3	5931	5574	94
Copy #4	5824	5570	96
Copy #5	6007	5579	93
Copy #6	5887	5568	95
Copy #7	5973	5570	93
Copy #8	5782	5548	96

Table 9-3. 8 on 4 Throughput Test

10.0 Conclusions

We have discussed the merits of multi-discipline analysis and examined performance of various MD Nastran solvers on both SGI Altix® IPF and XE. At this point in time, there is no clear performance winner. Even though XE always wins for small problems, and certain solver types, the scale would tilt toward IPF for very large problems and other types of solvers. A shared file system also provides a significant cost advantage for running IO intensive job mixes on IPF, since a high performance shared file system for clusters still would be a very expensive acquisition today. Therefore, the user should carefully examine all variables - performance, user needs, throughput requirements, price of the system, system administration work loads, power and cooling costs, and software license fees, etc - and select the computer system accordingly.

ACKNOWLEDGEMENTS

The authors would like to thank the following MSC staff in alphabetic order for making this study possible: Casey Heydari, Joe Griffin, Kevin Kilroy, Keith Leung, Peter Schartz, Paul Vanderwalt, Per Nordlund, Rob Ford, Sanjay Choudhry, Walter Schrauwen.

REFERENCES:

(1) Chinner, D. and Higdon, J., “Exploring High Bandwidth Filesystems on Large Systems”, Proceedings of the Linux Symposium, Volume One, July 19th-22nd, 2006, Ottawa, Ontario, Canada.

(2) SGI Propack man page for “DPLACE”.

(3) SGI Propack man page for “libFFIO”.

(4) MD R2 Nastran Release Guide, The MacNeal-Schwendler Corporation, Los Angeles, CA, 2007.

(5) Nastran Release Guide, v2001, The MacNeal-Schwendler Corporation, Los Angeles, CA, 2002.

(6) Nastran Release Guide, v2004r3, The MacNeal-Schwendler Corporation, Los Angeles, CA, 2004.

(7) Kodiyalam, S., Yang, J., and Gu, L., “Multidisciplinary Optimization of a Vehicle System in a Scalable, High Performance Computing Environment”, Structural and Multidisciplinary Optimization, Volume 26, pp. 256-263, 2004.

(8) SGI Propack man page for “PCPIntro”.

Corporate Office
1140 E. Arques Avenue
Sunnyvale, CA 94085
(650) 960-1980
www.sgi.com

North America +1 800.800.7441
Latin America +55 11.5185.2860
Europe +44 118.912.7500
Japan +81 3.5488.1811
Asia Pacific +61 2.9448.1463

4085 [03.07.2008] **J15363**