Considerations of MSC.Nastran Parallel Efficiency for Automotive NVH Applications

Cheng Liao, Ph.D., Principal Engineer, CAE Applications
SGI Mountain View, CA, liao@sgi.com, 650.933.3579

Stan Posey, Automotive Industry Market Development
SGI Mountain View, CA, sposey@sgi.com, 650.933.1689

Abstract

The automotive industry continues their increased investments in NVH analysis and as such, the number of MSC.Nastran users within major automotive companies has grown larger and demands more from computer system environments. These demands typically include rapid job turnaround and throughput capability in a multi-user HPC environment. What are more, many NVH analysis environments coexist with other structural software applications that compete for system resources such as CPU, memory, storage and I/O.

A discussion is given on the parallel NVH analysis techniques with MSC.Nastran solution sequences. Parameters such as model size, element types, and frequency cut-off value can produce a wide range of compute behavior such that consideration should be given to how system resources are configured. The parallel performance characteristics of the SGI 2000 and SGI 3000 are examined for both turnaround and throughput NVH requirements that include industrial-size automotive examples. In addition Simple guidelines on proper system configuration and innovative use of available IRIX system resource management tools are provided that are designed to maximize NVH analysis productivity.

1. Introduction

The use of NVH analysis provides essential benefits towards designing vehicles for ride comfort and quietness, an increasingly competitive advantage in today's global automotive market. Analysts continue to push NVH modeling to higher excitation frequencies in order to capture an increasingly larger audible range for additional NVH reductions. Subsequently this requires that NVH model parameters grow substantially larger and well beyond those in common modeling practice today.

Common global practice for NVH analysis on trimmed body-in-white (BIW) typically restricts upper bound limits on excitation frequencies to between 250Hz and 300Hz. However many automotive companies have desires to increase this to 600Hz and beyond during the next few years. These predicted modeling levels for higher frequencies couldn't be met with conventional NVH methods with regards to both numerical accuracy and practical job turn-around time.

The automotive industry has historically invested in vector systems to satisfy the high-performance computing (HPC) resource demands of CAE applications, and in particular for NVH analysis using MSC.Nastran. NVH requires high demand of virtually all HPC resources -- CPU, storage, memory bandwidth, and I/O rates on the order of TBS for a single NVH job. Parallel capability of NVH was introduced only recently, and vector systems were important for NVH jobs that were restricted to uniprocessor execution.

With release of MSC.Nastran v70.7 during 1999, the first substantial parallel capability for NVH analysis provided industry with a migration path from expensive vector systems to more economical RISC systems. A distributed memory parallel programming model is used for MSC.Nastran in order to leverage the advantage of contemporary scalable RISC architectures. This parallel capability was implemented for all NVH related solution sequences such as 103, 108, and 111.

Automotive industry migration in HPC architectures is the result of new algorithms and methods recently implemented for several automotive HPC applications. The vector-to-RISC migration began for the automotive industry during 1995 when a direct sparse solver was introduced in MSC.Nastran, and quickly became the choice over the expensive skyline solver. The sparse solver reduced CPU and storage requirements by an order of magnitude over the skyline solver and as such, structural analysis (static's) rapidly migrated from vector to RISC.

Lately there has been growing concern over how the industry will address future NVH modeling requirements at higher frequencies. Today's typical model sizes for current NVH analysis of an trimmed body requires a vector class system for effective job turn-around -- usually defined as overnight. With the current NVH techniques in place, even next generation vector systems will not deliver the performance required for conventional NVH modeling targets of the future.

Model parameters will exceed the practical limits of these vector architectures. The conventional eigenvalue extraction (Lanczos) and modal response methods (MSC.Nastran SOL 103 and 111) are widely considered to be too costly for vector, such that alternative methods are being proposed on cost-effective RISC systems. This paper examines efficient implementation of Lanczos on both uniprocessor and parallel RISC architectures and, a highly parallel direct frequency response (MSC.Nastran SOL 108) method as a consideration to achieve future NVH modeling targets.

2. MSC.Nastran for NVH Analysis

Implicit FEA software MSC.Nastran from developer MSC.Software Corporation (www.mscsoftware.com), is a multi-purpose structural analysis tool with a range of linear and nonlinear capabilities. It is most popular with industrial applications that require evaluation of the dynamic response. Of structures, For example, the automotive industry uses MSC.Nastran for design applications such as vehicle weight reductions, improved durability, and reductions in NVH.

Beginning with MSC.Nastran 70.7, a distributed memory parallel capability was developed with MPI used as the message passing software. NVH applications were the primary motivation for parallel MSC.Nastran since they require a more than 10-fold increase in compute resources over other applications. Body NVH in particular can require a day of processing on a vector system such as the Cray T90.

In general, two methods are used for frequency response evaluation in NVH applications; modal and direct. The modal method performs two steps; a Lanczos eigenvalue analysis that computes natural frequencies over an excitation range of interest, and later a frequency response based upon on the generalized modal coordinates from the Lanczos step. The direct method is a single step solution of the equations of motion for each frequency in the range of interest.

The modal method is conventional practice for practically every automotive NVH application today, owing to lower cost vs. the direct method at low frequencies near the 200 Hz level. However, a shift is underway towards increased use of the direct method since high levels of parallel scalability are observed that can provide faster turn-around over modal. MSC.Software Corporation developed parallel capability for both modal and direct methods, but the direct has a naturally parallel algorithm that exhibits much higher scalability than the modal method.

Parallel scalability for the modal method is model dependent since the parallel Lanczos scheme divides the frequency range equally among processors, then performs the eigenvalue search within each sub-range on each processor. For example, in a case where the cut-off frequency is 200 Hz for an 8 processor execution, processor 1 is assigned the frequency segment of 0 to 25 Hz, processor 2 from 26 to 50 Hz, and so on. Each processor then works independently to calculate the eigenvalues within their assigned segment.

This parallel Lanczos scheme can experience load-balance inefficiencies since it is rare for a structural to have an equal number of natural frequencies within each uniform frequency segment. The result is that some processors perform more work than others for a given segment. The direct method, however offers good load-balance since parallelization is implemented for each independent frequency step, which each perform roughly the same amount of work.

Parallel efficiency for NVH is a key consideration when using MSC.Nastran v70.7 and later releases, but it is important only after techniques for uniprocessor efficiency are implemented. The SGI ccNUMA architecture offers many features that can enhance MSC.Nastran performance for both uniprocessor and parallel execution. An examination of the various solution paths of MSC.Nastran help to explain the demands required of a particular hardware architecture feature.

Generally speaking, finite element software exhibits a range of compute behavior depending upon the kind of analysis being conducted and model size, such that balanced hardware architecture is desired. For NVH modeling, parameters such as the size of the model, the type of geometry, the types of elements, and the excitation frequency of interest, all affect the MSC.Nastran execution behavior. A profile is provided in Table 1. That describes the behavior for certain MSC.Nastran tasks associated with NVH analysis.

Table 1. Computational Profiles for MSC.Nastran and NVH Analysis

Computational Task	Memory Cycles	CPU Cycles
Sparse Direct Solver	7 %	93 %
Lanczos Solver	60 %	40 %
Iterative Solver	83 %	17 %
I/O Activity	100 %	0 %

This profile highlights the importance of a balanced system and consideration for uniprocessor efficiency, since the sparse direct solver requires a fast processor for effective execution while the Lanczos solver requires high memory bandwidth speeds. The requirement for bandwidth is even greater when considering jobs that require I/O or use of the iterative solver. The I/O requirement in particular is very critical to efficiency in elapsed time for NVH since Lanczos is highly dependent upon large amounts of I/O for models with high modal density such as those typical of body NVH modeling.

3. SGI ccNUMA Architecture

The MSC.Nastran computations presented in this paper were performed on the SGI 2000 (formerly Origin2000) and SGI 3000 family of servers. These are cache-coherent non-uniform memory access multiprocessor (ccNUMA) architectures [1]. This ccNUMA architecture is a breakthrough implementation of shared memory architectures. For the SGI ccNUMA design, memory is physically distributed among the nodes but it is globally addressable to all processors through an interconnection network.

The motivation and direction towards ccNUMA evolved at SGI as traditional shared bus architectures like that of the CHALLENGE server began to exhibit high latency bottlenecks as processor counts were growing within a single system image. During this same time, non-coherent distributed memory architectures started to emerge, but the programming of applications for message passing in such an environment was considered too difficult for commercial success.

The SGI ccNUMA implementation distributes memory to individual processors through a non-blocking interconnect design, in order to reduce latencies that inhibit high bandwidth and scalability. At the same time, a unique directory based cache-coherence provides a memory resource that is globally addressable by the user, in order to simplify programming tasks. A single image SGI 2800 system offers up to 512 processors and can expand to 1 Tbyte of memory, which is the largest SMP system currently available in industry.

The distribution of memory among processors ensures that memory latency is reduced. The globally addressable memory model is retained but memory access times are no longer uniform. The ccNUMA design incorporates hardware and software features that minimize latency differences between remote and local memory. Page migration hardware moves data closer into memory locations that are closer to a processor with frequently access to that data, meaning that most memory references are local.

Cache coherence is maintained via a directory-based protocol while caches are used to reduce memory latency as well. While data only exists in either local or remote memory, a copy of the data can exist in various processor caches. Keeping these copies consistent is the responsibility of the logic (cache-coherent protocol) of the various hubs. The directory-based cache coherence protocol is preferable to snooping since it reduces the amount of coherence traffic; cache-line invalidations are broadcasted only to those processors actually using the cache line instead to all processors in the system.

The building block of the system is the node, which contains two processors for the SGI 2000 and four processors for the SGI 3000, and up to 4 GB of main memory. The node also contains its corresponding directory memory, and a connection to a portion of IO subsystem. The hub interface to the node is the distributed memory controller and is responsible for providing transparent access to all of the distributed memory (in a cache-coherent manner) to all of the processors and I/O devices. The nodes can be connected together via any choice of scalable interconnection network.

System architecture's ability to achieve high parallel efficiency becomes increasingly important as algorithms for CAE software applications are developed towards such capability. From a hardware and software algorithm perspective, there are roughly three types of CAE simulation "behavior" to consider: implicit and explicit finite element analysis (FEA) for structural mechanics, and CFD for fluid mechanics. Each has their inherent complexities with regards to efficient parallel scaling, depending upon the parallel scheme of choice.

Most commercial CAE software employs a distributed-memory parallel (DMP) implementation for compatibility across the full range of RISC architectures available. Other choices for efficient parallel methods are shared-memory parallel (SMP) coarse grain, and hybrid parallel schemes that combine DMP and SMP within a single simulation. MSC.Nastran uses a DMP implementation with MPI but also contains SMP for certain tasks such as element generation and assembly. This strategy is well suited to distribute shared-memory architecture like the SGI ccNUMA.

4. Parallel Performance Issues

Parallel efficiency of any CAE software application has certain algorithm considerations that must be addressed. The fundamental issues behind parallel algorithm design are well understood and described in various research publications. For grid-based problems such as the numerical solution of partial differential equations, there are four main sources of overhead that can degrade parallel performance:

*** Non-optimal Algorithm Overhead**
The best sequential algorithm may often be difficult or impossible to parallels, for example triangular solvers. In such cases the parallel algorithm may have a larger operation count than a sequential one. Additionally, in order to avoid communication overhead one may wish to duplicate some computations on different processors.

*** System Software Overhead**
Implementation of a parallel algorithm often results in an increase of the (relative) software overhead such as those associated with indexing, procedure calls, and other operations that are not floating-point computations.

*** Computational Load Imbalance**
The execution time of a parallel algorithm is determined by the execution of the processor performing the largest amount of work. Should the computational workload not be evenly distributed, load imbalance will result and processor idling will occur, meaning certain processors must wait for other processors to finish their computation.

*** Communication Overhead**
All time spent in communication and synchronization through both hardware and software means is pure overhead. An example is the communication latency associated with the explicit passing of messages in a DMP scheme.

The parallel version of MSC.Nastran was carefully designed to minimize major sources of parallel inefficiencies. For certain solution sequences such as 108, communication overhead is minimized and proper load balance is achieved. Also, implementation of an SGI ccNUMA-aware MPI helps to minimize communication overhead since latency is reduced by about 3-fold over MPICH. Still with such a sophisticated approach, parallel performance can exhibit unsatisfactory results on a ccNUMA system owing to the lack of special mapping to the specific architecture features of ccNUMA.

Parallel performance of MSC.Nastran, as originally ported on the SGI ccNUMA architecture did not meet initial expectations, with most models scaling only to 2 or 4 processors. Examination of the parallel performance for a number of cases identified bottlenecks with MPI latency and non-enforcement of processor-memory affinity (data placement) as the key reasons for limited scalability. The data placement concern was related to a feature of the ccNUMA architecture and was addressed through implementation of the IRIX dplace set of tools.

Total MPI latency is determined by both the specifics of system architecture and the implementation of MPI for that system. Since system architecture latency is determined by design of a particular interconnect, overall latency improvements can only be made to the MPI implementation. Modifications to the MPI software to ensure "awareness" of a specific architecture are the only way to reduce the total latency and subsequently the communication overhead.

5. MSC.Nastran Computational Efficiency

The overall objective of the following performance studies is to demonstrate techniques for improved performance of MSC.Nastran software on the SGI ccNUMA architecture for a range of solution sequences related to NVH analysis. Performance is examined on moderately configured SGI 2000 and 3000 systems that are described in Table 2.

Table 2. System Environments for MSC.Nastran Performance Investigations

System	Processor	Level2 Cache	System Topology	IRIX Level	MPI Latency (microsec)
SGI 2400	MIPS R12000/300Mhz	4 MB	1 x 32	6.5	11.8
SGI 2400	MIPS R12000/400Mhz	8 MB	1 x 16	6.5	11.8
SGI 3200	MIPS R12000/400Mhz	8 MB	1 x 8	6.5	6.2

A review of techniques for improved uniprocessor efficiency is provided, prior to discussion about multi-processor efficiency considerations. Note that these techniques are valid for both homogenous and heterogeneous CAE application environments, meaning there would be no impact on jobs from other applications executing simultaneously with MSC.Nastran from the following recommendations.

5.1 Uniprocessor Efficiency

Efficient NVH performance from a uniprocessor perspective demands high rates of memory bandwidth as noted for the Lanczos algorithm from Table 1. in addition to high rates of I/O bandwidth. A key feature of the SGI 2000 ccNUMA architecture is its memory bandwidth rate of 720 MB/sec shared by two processors on a single node. For the SGI 3000 it is 3200 MB/sec shared by four processors on a single node. Since an estimated 60% of all compute cycles for the Lanczos algorithm are associated with memory traffic, significantly faster processors will not produce significant increases in performance.

Improvements in uniprocessor Lanczos performance for modal frequency response, and therefore SOL 103 and 111 jobs, will come from increases in memory and I/O bandwidth. Also noted from Table 1. is that requirement for uniprocessor improvement in the sparse solver is all about processor speed rather than memory and I/O bandwidth. Since direct frequency response is dominated by sparse decomposition, the increase in bandwidth offered by the SGI 3000 over the SGI 2000 will not offer much of an increase in uniprocessor performance for SOL 108.

To better illustrate NVH uniprocessor efficiency and execution behavior, a series of MSC.Nastran models from various solution sequences were selected that exhibit a range of system resource requirements. Each of these models represent industrial sized models that appropriately stress the resources of a complete system environment. The models are shown in Table 3. and provide details on the solution sequence and size of each, along with other NVH parameters where appropriate.

Table 3. MSC.Nastran Models for Uniprocessor Performance Investigations

Case	SOL	DOF	Modes	Frequencies	Design Loops
A	103	650K	814	n/a	n/a
B	111	1067K	660	100	n/a
C	108	525K	n/a	4	n/a
D	200	570K	n/a	n/a	2

Each of the models from Table 3 was executed on each of the systems described in Table 2. In each case steps were taken to ensure maximum uniprocessor efficiency. As stated in section 4. it is important to ensure a good mapping of the memory placement of data with the process associated with that data. This is accomplished with enforcement of memory-processor affinity through a set of SGI system tools called "dplace" for data placement tools. It should be noted that MSC.Nastran is instrumented for this feature on the SGI ccNUMA systems so that this requirement is transparent to the user.

There are certain MSC.Nastran system parameters that can benefit NVH and other applications. For the SGI ccNUMA architecture it is recommended that the following parameters be considered, which can be set overall in the "nastrc" resource file:

SYSTEM (205) = 64, SYSTEM (198) = 8
SYSTEM (206) = 4
SYSTEM (273) = 2
SYSTEM (1) = 65537

The parameter setting for the sparse solver RANK size (205) and the Lanczos MINFRONT size (198) provides good performance for most MSC.Nastran modal NVH jobs for the MIPS R12000 processor. For sparse solver dominant jobs, one may wish to try various RANK size parameters (other than 64) for improved performance. A model with a large number of DOF and small numbers of modes is sparse solver dominant, which are characteristics typical of powertrain NVH.

Also recommended for the sparse solver is EXTREME reordering which is selected with 206. Selection of the proper reordering scheme for a particular model can increase the reduction of floating point operations in the sparse decomposition. Other reordering methods that have shown improvement for the SGI architecture beyond the default reordering include METIS (206 = 8) or MMD (206 = 2).

For modal NVH performance improvements, there is an SGI ccNUMA specific Lanczos shifting logic to maximize eigen-extraction performance for large numbers of modes, typically several hundred or more. This option is enforced with the 273 = 2 parameter. Other choices include the default (273 = 0) and the BCS shifting logic (273 = 1) which may be better methods when the number of modes is very small. The improvements from 273 are all highly model dependent.

Also important for large numbers of modes is proper I/O configuration. The parameter 1 = 65537 provides the maximum BUFFSIZE allowed by the system, which is the first step. Most important is configuration for a large scratch file system and use of the SGI EAG_FFIO library through the parameter **ff_io_opts**. Reference to [5] has details on the use of this parameter, but the following example shows set up of an FFIO cache of 512 x 1 MB pages and 4 read "aheads" for the MSC.Nastran SCRATCH and SCR300 files:

ff_io_opts = '*.SCR*(eie.direct.diag.mbytes:256:512:4:1:1:64,event.summary.mbytes)

Performance results are provided in Table 4. for the uniprocessor cases described in Table 3. for each of the SGI systems described in Table 2. For case A which is dominated by the Lanczos algorithm owing to a large number of modes relative to DOF, very little improvement is observed with an increase in the MIPS clock speed (300 Mhz to 400 Mhz) yet very good improvement is observed between the SGI 2000 and SGI 3000 architectures.

The contributions of increased processor speed and L2 cache size are observed as expected with case C which is dominated by sparse decomposition, and case B which experiences a mix of dependence on Lanczos and sparse decomposition algorithms, given the large DOF and the relatively moderate number of modes. For case D which is a design optimization model, the iterative scheme behind SOL 200 benefits from the improved bandwidth of the SGI 3000 architecture.

Table 4. MSC.Nastran Uniprocessor Performance

Case	SOL	SGI 2400-300	SGI 2400-400	SGI 3200	3200 / 2400-300
A	103	20,638 s	0,630 s	13,870 s	1.49 x
B	111	41,180 s	29,043 s	23,755 s	1.73 x
C	108	8,024 s	6,276 s	5,931 s	1.35 x
D	200	62,465 s	52,292 s	43,329 s	1.44 x

The ratio given between the SGI 2400-300Mhz system and the SGI 3200-400Mhz system show a range of 35% to 73% improvement depending on the model parameters. These uniprocessor improvements were obtained from a combination of efficient system parameter considerations, an increase in MIPS 12000 processor speed, and improved bandwidth of the SGI ccNUMA system architecture. Considerations for MSC.Nastran parallel efficiency should be examined only after a good knowledge of uniprocessor efficiency is achieved.

5.2 Parallel Efficiency

A discussion on the distributed memory parallel implementations of modal and direct frequency response using MPI were provided in section 2. MSC.Nastran was originally developed as a shared memory parallel implementation but the MPI version gives much improved parallel scalability. One trade-off however, is that the MPI implementation also requires more system resources over the SMP.

The uniprocessor issues were given and parallel efficiency is examined. Examples are provided that demonstrate the MPI parallel efficiency of the most important solution sequences for NVH -- SOL 103, 111, and 108. For each model the data placement issues described in section 4. were resolved through automatic configuration in the MSC installation that is user transparent.

The first example illustrates SOL 103 parallel performance for a vehicle body that is typical of the automotive industry today. The results in Table 5. show that reasonable scalability is achieved up to 4 processors. Limitations on the parallel Lanczos algorithm were described in section 2. The value in the SGI 3000 high bandwidth architecture is also illustrated since the SGI 2400-300 result of about 17 hours (16,595s) is reduced by 35% to about 11 hours on the SGI 3200 single processor, and is further reduced to just 3.5 hours using 8 processors.

Table 5. MSC.Nastran SOL 103 Parallel Performance for xxcmd Model

Model: Auto body, 1.3M DOF, 1088 modes, 800 MB memory, 2.4 TB total I/O
Metric: Elapsed seconds with parallel speed-up

CPUs	SGI 2400-300	SGI 2400-400	SGI 3200
1	61,595 *1.0 **	55,724 *1.0 **	40,523 *1.0 **
2	36,734 *1.7*	33,453 *1.7*	23,831 *1.7*
4	24,501 *2.5*	22,218 *2.5*	15,582 *2.6*
8	19,474 *3.2*	17,497 *3.2*	12,276 *3.3*
16	14,660 *4.2*	n/a	n/a

* All speed-up results are relative to the 1 CPU result

The second example illustrates SOL 111 parallel performance for a vehicle body with a small number of modes but a large number of frequencies. As expected, parallel speed-up of the modal frequency response method, which includes a Lanczos eigenvalue extraction shows less parallel efficiency than SOL 103. Parallel speed-up is not very encouraging but the gains provided by the SGI 3000 high bandwidth architecture help to overcome some of the absolute performance issues. About a 42% decrease in turn around is observed between the SGI 2400-300 time of about 4 hours to about 2.5 hours on the SGI 3200 single processor.

Table 6. MSC.Nastran SOL 111 Parallel Performance for xlemf Model

Model: Auto body, 660K DOF, 39 modes, 449 freq, 400 MB memory, 474 GB total I/O
Metric: Elapsed seconds with parallel speed-up

CPUs	SGI 2400-300	SGI 2400-400	SGI 3200
1	15,148 *1.0 **	10,820 *1.0 **	8,872 *1.0 **
2	12,619 *1.2*	9,146 *1.2*	7,397 *1.2*
4	9,463 *1.6*	6,851 *1.6*	5,549 *1.6*
8	8,421 *1.8*	5,864 *1.8*	4,931 *1.8*

*All speed-up results are relative to the 1 CPU result

The third example examines parallel performance of SOL 108 for a vehicle body with 96 frequencies. A discussion of the parallel implementation of the direct frequency response algorithm was provided in section 2. Table 7. shows SOL 108 results that demonstrate high parallel efficiency for a large number of processors. Results on the SGI 2400 from 1 to 32 processors produced a 23-fold speed-up, which reduced the turn-around time from 31.7 hours to only 1.4 hours. On 16 processors the speed-up was 14-fold, demonstrating very good efficiency.

Table 7. MSC.Nastran SOL 108 Parallel Performance for xlifr Model

Model: Auto body, 536K DOF, 96 frequencies
Metric: Elapsed seconds with parallel speed-up

CPUs	SGI 2400-300		SGI 3200	
1	114,264	*1.0* *	84,640	*1.0* *
4	28,887	*4.0*	21,168	*4.0*
8	14,883	*7.7*	10,855	*7.8*
16	8,070	*14.7*	n/a	
32	5,047	*22.6*	n/a	

* All speed-up results are relative to the 1 CPU result

These result provided in Table 7. are very encouraging, such that SOL 108 can be considered as an alternative to the less efficient Lanczos parallel method, and especially given that SOL 108 provides improved solution accuracy. Higher frequencies require that NVH model parameters grow substantially larger from current practice and a highly parallel SOL 108 is a potential method to achieve these goals.

6. Future Directions

Significant NVH analysis improvements have occurred recently such that further NVH reductions will only occur with design investigation at higher levels of fidelity and precision. An NVH capability that is highly scalable is critical to the automotive industry as NVH modeling expands to higher excitation frequencies that are necessary for modeling within a higher audible range. Further noise reductions for the occupant are possible with a better understanding of noise sources at these higher frequencies.

Higher frequencies require that NVH model parameters grow substantially larger from current practice. NVH models today for trimmed BIW are typically limited to frequencies of 250 to 300 Hz, yet many automotive companies want an increase to 600 Hz. The conventional modal method requires vector systems for overnight turn-around, but even next generation vector designs will not deliver the performance required for future targets of NVH modeling. Also, accuracy concerns for the modal method at higher frequencies do not exist for the direct method.

The automotive industry "standard" for NVH has been the modal method and vector systems for several years. It will be necessary to characterize the crossover point whereby the parallel direct method with RISC systems becomes less expensive computationally. A third example demonstrates this with a comparison of a vehicle BIW using the direct method on the same SGI 2800/64, and the conventional modal method on a single processor Cray T90. The model contains 525K DOF and extracts 2714 modes for a response analysis of 96 frequency steps.

Results demonstrate roughly equal performance between the two techniques when 4 processors are used for the SGI 2800. At this level, the Cray T90 time is 8.8 hours compared with the SGI 2800, 4 processors at 8.9 hours. The 4 processor time exhibits near-ideal speed-up at 3.8-fold over 1 processor at 33.5 hours. Results for the SGI 2800 at 8 processors further reduces the turn-around time to 4.8 hours for a 7-fold speed-up over that for 1 processor.

In the wake of recent breakthroughs for scalable CAE simulation, research and industry will continue to increase their investments in CAE technology as a product and process design aid. The motivation is

simply a matter of economic benefits and improved quality that scalable CAE brings to the development process. Efficient turn-around of CAE simulations means increased modeling resolution and more comprehensive evaluation during early development stages. Parallel NVH analysis with MSC.Nastran is a technology that will contribute to that trend.

Advancements will continue well into the new decade for improved CAE scalability as emerging algorithm developments and new hardware architectures lead a path towards enhanced CAE methodologies. These enhancements will encourage an increase in CFD modeling for transient flow conditions, widespread implementation of probabilistic structural mechanics, and production capability for multi-discipline fluid and structure coupling, among others.

Future modeling requirements for trimmed body NVH are not practical with the conventional eigenvalue extraction and modal response method currently in practice. A highly parallel direct frequency response method is a viable alternative that offers practical job turn-around time, and with equivalent-or-better numerical accuracy. Use of the direct frequency response method also offers the automotive industry a migration path from vector to more cost-effective scalable RISC. The low cost of RISC computing will also enable rapid growth of design optimization.

7. Conclusions

Presented were guidelines for efficient implementation of MSC.Nastran for NVH applications on the SGI ccNUMA architecture. Modal frequency response with the Lanczos algorithm exhibits efficient performance when considerations are give to data placement and proper I/O configuration. The Lanczos algorithm in particular requires special attention to architecture awareness parameters.

The highly scalable direct method frequency response offer an efficient alternative to Lanczos for high frequency applications. It was shown that modal density increases nonlinearly with increasing excitation frequency, meaning the direct method offers an encouraging alternative to conventional vector-based NVH modeling, based upon both performance, and cost-performance. An additional benefit is the improved solution accuracy of the direct over the modal method.

References

[1] Laudon, J. and Lenoski, D., "The SGI Origin ccNUMA Highly Scalable Server," SGI Published White Paper, March 1997.

[2] Komzsik, L., MSC.Nastran Numerical Methods, Version 70, 1998.

[3] Dagum, L., McDonald, J., and Menon, R., "The NAS LU Benchmark as a Scalable Shared-Memory Program", Silicon Graphics Internal Report, 1997.

[4] Duncan, A., Su, F., and Wolf, W., "Understanding NVH Basics." Proceedings of the International Body Engineering Conference. Body Design and Engineering, Vol. 20, pp. 111-116, 1996.

[5] EAG_FFIO Programming and Usage Guide, http://www-devtoolbox.engr.sgi.com/toolbox/src/HPC/io/eag_ffio