

ANSYS Benchmarking Project: Evaluation of the Distributed Domain Solver

David O'Neal

National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign

Sam Murgie

Technical Fellow
ANSYS, Inc.

Abstract

The focus of this paper is the *Distributed Domain Solver* (DDS), a prominent feature of the *Parallel Performance for ANSYS* (PPFA) add-on product introduced in 2001. The DDS program is based on the method of *finite element tearing and interconnecting* (FETI) developed by Farhat and Roux in the early '90s for an aerodynamic study of the high-speed civil transport aircraft.ⁱ A distinguishing attribute of the FETI algorithm is that it requires fewer interprocessor data communications than traditional domain decomposition algorithms. Convergence properties and scalability of the research code were demonstrated by Farhat, Mandel, and Roux in 1994.ⁱⁱ In this report, we examine resource utilization and scalability of the industrial application operating in the field.

An extensible input model was created for the project, and then cases ranging from about 60,000 to over ten million *degrees-of-freedom* (DOF) were defined for testing purposes. A large Silicon Graphics Origin 2000 system and a commodity cluster of dual-processor Pentium III workstations were selected to perform the experiments. The Windows/NT and IRIX production releases of ANSYS 5.7 were installed, and testing commenced.

Core sections of experimental data follow reviews of relevant system characteristics. Profiling details were fully developed for possible application within the context of the ANSYS e-CAE™ portal service. An effort was made to introduce key observations in proximity to the charts from which they were derived. Trailing remarks for these sections consist primarily of reflections on these points. Final comparisons of DDS scalability are made with respect to both test platforms, and a few key recommendations are presented in closing.

Introduction

With the release of the Distributed Domain Solver (DDS) in January 2001, ANSYS has emerged as a leader in commercial applications development for distributed computing environments. A combination of three characteristics sets DDS apart from other parallel equation solvers:

- Distributed memory design based on Message Passing Interface software (MPI)
- Minimal data communication requirements
- Optimal convergence properties

Implications of these statements are: DDS is adaptable to virtually any multiprocessor computing system, its efficiency is not heavily dependent on the presence of state-of-the-art networking hardware, and the global rate of convergence is independent of the subdomain count.ⁱⁱⁱ A pair of common cluster computers with very different networking characteristics was selected to test these assertions.

DDS evolved from an academic project in solid mechanics as an alternative to other parallel iterative methods based on conjugate gradient methods. Additional publications describing the method of finite element tearing and interconnecting (FETI) may be viewed by following the link indicated in the reference section of this document.^{iv}

An extensible model was designed to facilitate the application of many different mesh configurations to a single problem, but it also proved to be quite useful in another way as we quickly discovered that the initial production codes were unable to process some of the larger test cases. The ability to specify any mesh density allowed us to approach these operational boundaries gradually. When coupled with a binary search pattern, we were able to approximate the points at which problems were occurring with much greater accuracy. A version of this model may be obtained by following the indicated link.^v

The initial release of the Distributed Domain Solver imposes restrictions on the types of problems to which it may be applied. DDS is primarily intended for large static and full transient analyses represented by symmetric matrices. The following features are currently incompatible with DDS: pre-stress, inertia relief, coupling, constraint equations, LINK elements, p-elements, super elements, PRETS179 elements, and probabilistic design models. However, enhancements planned for 2002 include support for the handling of coupled systems and constraint equations.

The availability of the PPFA product may conjure up images of previously intractable industrial models running on supercomputing-class platforms, but its effects are further reaching than this. Improving time to solution promotes development of increasingly sophisticated models at every level. Better accuracy leads to better designs, which implies a competitive edge.

However, the most significant aspect of PPFA is that it embraces grid computing. While we don't expect large co-located resource centers to vanish any time soon, advancements in networking performance will erode their relevance. A small rack of desktop units represents a viable alternative to a high-capacity (read *expensive*) server, and the economics dictate that this trend will continue for some time.

Conventions

Although this article does not incorporate many symbols or units of measurement, a few conventions are followed that are worthy of mention:

- Acronyms are fully described in proximity to their first occurrence
- New technical terms are generally introduced with an *italic font*
- System commands and ANSYS keywords are displayed with a **fixed font**

All timings are reported in units of seconds and, unless noted to the contrary, all references to memory usages are given in megabytes (Mbytes).

Software

The ANSYS Distributed Domain Solver is comprised of a stand-alone executable distinct from the finite element analysis (FEA) host application. Control of DDS is affected by the host through a number of subtle modifications to the command set. For example, when the DDS option to the EQSLV command is referenced, a sequence of related tasks is initiated before DDS is launched.

The meshed model is first decomposed for parallel execution by a component derived from the METIS graph partitioning application.^{vi} Next, a file (.erot) containing the entire set of element matrices complete with domain-specific information is created. This file serves as the primary communication channel for transmitting data from the host program to DDS. At approximately one Kbyte per DOF, the size of this file can grow very quickly. By default, it is deleted immediately after completing the transfer.

Decomposition of the meshed model is generally accomplished with default settings, but a few documented options and a number of developer flags have been carried through from earlier implementations. The interface for the DDS options command looks like:

```
DDSOPT,ConfigOpt,Ndomains
```

where the ConfigOpt field describes platform-specific MPI launch criteria (see below) and the NDomains field is used to coerce a specific subdomain count out of the decomposition process.

The use of the default setting (AUTO) for the NDomains field is highly recommended, but for the adventurous, guidelines for specification of a count are: the number of subdomains must exceed the number of available processors, and the number of DOFs per subdomain should be between 1,00 and 2,000.

During the solution phase, DDS is launched by the host onto all designated compute nodes simultaneously. The host then simply waits for MPI process 0 to return a solution estimate. MPI process 0 is also responsible for brokering the distribution of the communications file, and thus representing a bottleneck that will be addressed later. After distribution of the data file has been completed, a preconditioner is applied to the domain interface problem and an iterative sequence of *conjugate projected gradient* transformations is applied concurrently to each subdomain until convergence criteria is met.

Upon completion of the computational phase, communication of data associated with a residual norm on the projection space is passed between neighboring processes. The cycle continues until one of three conditions is met: the residual norm reaches a specified tolerance limit, acceleration of convergence vanishes, or the limiting count is reached. Solution data and diagnostics are then relayed back to the host via MPI process 0 and DDS terminates.

DDS is a message passing executable, so it must be started by an MPI job launcher. The implication is that a platform-specific launch command string must be embedded in the host executable. Control over this string is provided in the first field of the DDSOPT command. Usage details appear in the Experimental Notes section. The Windows/NT product uses the interface supported by MPI Software Technology's MPI/Pro package.^{vii} The IRIX port presumes the naming convention of the SGI Message Passing Toolkit.^{viii}

Some DDS executables are also threads-enabled. Available versions are based on OpenMP directives. These *hybrid* codes allow any combination of processes and threads to be specified. Conservation of memory, not performance, is the primary reason to favor the use of the threads model. Data replications are minimized. Availability of a Windows 2000 OpenMP compiler^{ix} has recently made it possible to develop a threaded Windows product, but at the time of this writing, it had not yet been released for testing.

Three new DDS executables for each operating system were provided and tested over the course of this project: a single processor performance upgrade and two bug-fix releases. Replacements for the original DDS production codes were provided early on. The change was driven by availability of a new high performance version of DGEMM (double precision matrix-matrix multiply) that had been integrated with

the Intel Math Kernel Library during the first quarter of 2001. Windows/NT timings showed a 25% improvement.

We first observed a resource management problem in the Window/NT executable. An apparent memory leak was limiting the range of problem sizes to around 500K DOF. It took a few months to isolate the bug and make repairs, but the changes ultimately resulted in the ability to handle problems consisting of over 1M DOF with only eight compute nodes. Note, however, that the node responsible for running MPI process 0 still carries a memory overhead that ultimately restricts the maximum problem size that can be solved on a cluster of nodes with uniform memory spaces. Therefore, Windows clusters designed to run ANSYS DDS should include at least one compute node with additional memory.

A second problem arose after we had already begun to collect experimental data. The IRIX executable failed for medium-sized problems (250K DOF) when more than a single OpenMP thread was specified. Multiple MPI processes worked fine. The problem was easily identified, as DDS returned an invalid error norm and, upon closer inspection, incorrect results. A new executable was provided within a month and testing continued.

All final timings were generated with the pair of executables provided by ANSYS in May 2001 that incorporated all of the modifications described above.

Systems

Two relatively common computing platforms were selected for this evaluation: a commodity cluster of Windows/NT machines and an SGI/Cray Origin 2000 system.

We were initially provided with exclusive access to a set of four Compaq 6000 Professional Workstations running Windows/NT. Each node featured dual 333-MHz Intel Pentium II processors and 512 Mbytes memory. A single front-end machine was used throughout the project to run the application and launch the DDS executable. All project software was installed and maintained on a shared network volume. All nodes were running Terminal Server. Service Pack 6 had been applied everywhere as required by MPI/Pro 1.6.

This very accessible system provided us with a versatile working environment with which to experiment prior to dedicating production resources to the project. It was also instrumental in the characterization of a memory leak that had plagued early releases of the Windows/NT DDS executable. Because we were able to login on every node while executions were in progress, usage statistics (commit charges, physical memory usage, and CPU utilization) could be closely monitored via Task Manager.

A second Windows/NT test configuration comprised of eight Hewlett-Packard Kayak XU Workstations was used to produce the timing data and utilization statistics presented. These systems featured twin 550-MHz Intel Pentium III processors and 1 Gbyte of memory each. Support for Myrinet hardware wasn't available from MPI Software Technology and so all runs were performed with the standard MPI/Pro Ethernet drivers (100 Mbit/s).

At the time these experiments were performed, the NCSA Origin 2000 cluster consisted of 12 machines and 1520 processors. Two of these machines (balder and forseti1) were used to generate all IRIX data presented herein. The targeted systems were populated with 250 MHz MIPS R10000 processors (256 and 128 of them, respectively) and 1 Gbyte of memory per node (dual-processor boards). Both machines were running IRIX 6.5.

The NCSA Origin cluster also featured Platform Computing's Load Sharing Facility^x batch system software and the Maui job scheduler.^{xi} A set of dedicated queues was used to feed jobs to the machines referenced above. All experimental runs were executed out of these queues.

Charts

Windows/NT experiments were limited to single-thread, multiple process runs on a relatively small set of partition configurations. This was the result of the unavailability of a threads-enabled executable, and limitations imposed by the MPI/Pro demonstration license.

In the next section, elapsed time and cumulative memory usage details are presented in pairs of bar charts. A summary of results associated with the set of the Windows/NT examples is also presented to give indication of the quality of DDS scalability with respect to an entirely affordable cluster system.

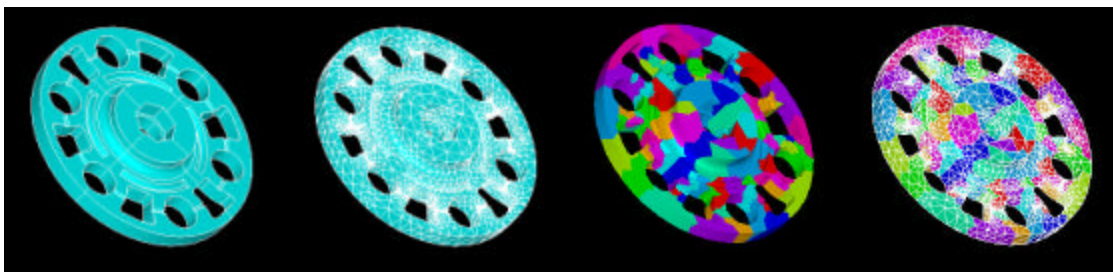
Sets of three charts are presented for each of the IRIX test cases. All process-thread combinations are included for each of the examples. Elapsed timings and memory usages for the DDS executable, and a third chart reflecting total CPU time consumed by each job are given. DDS statistics were taken from output produced by the solver executable, and the total CPU times were taken from accounting data reported by LSF. The inverse relationship between elapsed time and cumulative CPU time is clearly illustrated.

In all bar charts, floor axes correspond to process counts. For Windows/NT, the number of nodes and processes-per-node are considered. For IRIX, it's MPI processes and the number of OpenMP threads. One other common characteristic is that all data series have been defined with respect to their process-per-node (Windows/NT) or OpenMP thread (IRIX) counts. For example, all usages associated with a single process-per-node/thread are the same color (periwinkle).

Experiments

All computational experiments described herein are variants of the so-called *carrier model* created by ANSYS, Inc. The modeled part appears to be a component of a power transmission coupling, but in fact, it was synthesized specifically for DDS testing and has no physical significance.

All test cases were discretized using the volume meshing (VMESH) command. Element sizing for examples c01 through c10 was controlled through manipulation of the SIZLVL argument to the SMRTSIZE automatic mesh generator option. Larger problems (c20, c34, c72, and c87) were defined through specification of a maximum element diameter (DESIZE).



Carrier Model

The images above illustrate a typical discretization and decomposition of the carrier part. Area and element plots are included. The latter pair has also been decomposed and colored with respect to domain (/PNUM, DDS).

Host execution parameters were applied consistently in the following manner.

```
Examples c01 through c10: ansys57 - pp - m 1000 - db 500
Examples c20 through c87: ansys57 - pp - m 16000 - db 4000
```

The first argument (- pp) enables the use of Parallel Performance for ANSYS licensing. It is a prerequisite for running DDS. The others are *workspace* specifications that control the way in which memory is

allocated to a job. These settings only affect the performance of the host program, but of course, both components contribute to the time to solution.

The workspace option (`- m`) determines the size of the initial static allocation, as well as any subsequent dynamic allocations. Workspace is comprised of two components: *database* and *scratch* memory. Scratch space is implicitly defined as the difference between the workspace and database (`- db`) settings. Values are given in megabytes.

The ANSYS host is capable of automatically expanding workspace dynamically, but our tests indicated that specification of a sufficiently large initial allocation resulted in better turnaround times. Total CPU remained constant, but significant reductions in wall times were observed. These observations are consistent with the elimination of virtualized memory operations.

Identical input files were used to drive the Windows/NT and Origin analyses. Nevertheless, small differences in the element, node, and DOF counts were immediately apparent. Presumably, they were the result of differences in floating point implementations. Minor discrepancies between numerical results were not unexpected.

In general, ANSYS input files require only two modifications to select and run DDS:

```
EQSLV,DDS
DDSOPT,FILE
```

The first line chooses the domain decomposition solver. An optional convergence tolerance may be specified (default value is 1.e-6). The second describes the way in which the DDS executable is to be launched. The `FILE` option indicates that an MPI process group file is to be used to configure the `mpirun` command. The default filename is `config57.dds`.

The format of the process group file is determined by operating system. For the IRIX executable, a single line is usually sufficient:

```
- np process_count /path/ansdds.e57 - n thread_count
```

while the Windows/NT process group file uses a host list format, e.g.

```
\\hostname0\path\ansdds.exe - n process_count
\\hostname1\path\ansdds.exe - n process_count
```

Platform-specific details regarding the use of MPI process group files generally appear in the manual pages for the `mpirun` command, or variations thereof, for the targeted system.

Windows/NT Results

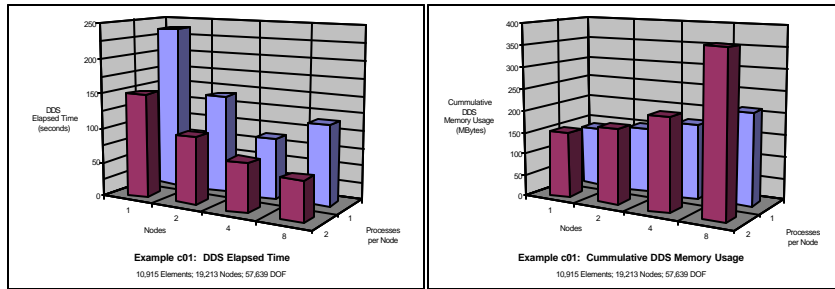
In following charts, DDS elapsed times and cumulative memory usages for tests involving one and two processes per node are presented.

Memory usage for equal numbers of processes are necessarily equal because the NT executable is not threaded. No exceptions were observed. Note that equal process counts always occur diagonally in these charts.

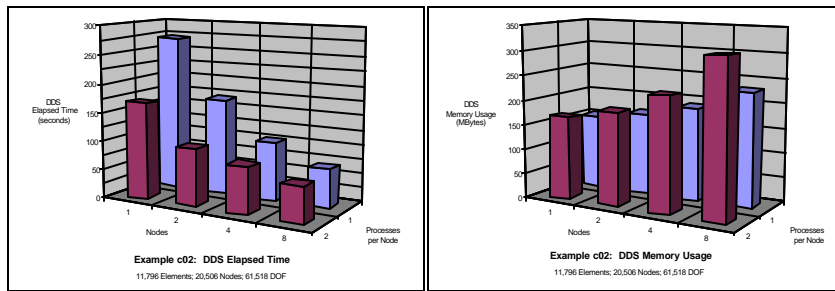
Larger problems could not be tested on configurations consisting of only one or two nodes due to insufficient memory, and so the range of partition sizes was adjusted to suit the problem:

- c01 through c06: 1, 2, 4, and 8 nodes
- c07 and c08: 2, 4, 6, and 8 nodes
- c09 and c10: 4, 6, and 8 nodes

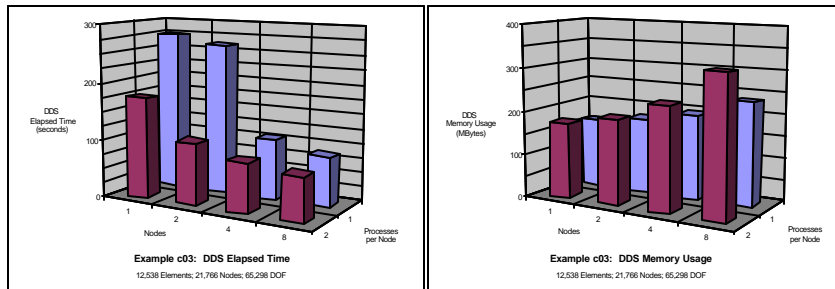
All readings were extracted from the DDS log file (jobname.DCS) and all unexpected results were reproduced a minimum of three times. Best observed times are cited.



Example c01 - 10,915 Elements • 19,213 Nodes • 57,639 DOF



Example c02 - 11,796 Elements • 20,506 Nodes • 61,518 DOF



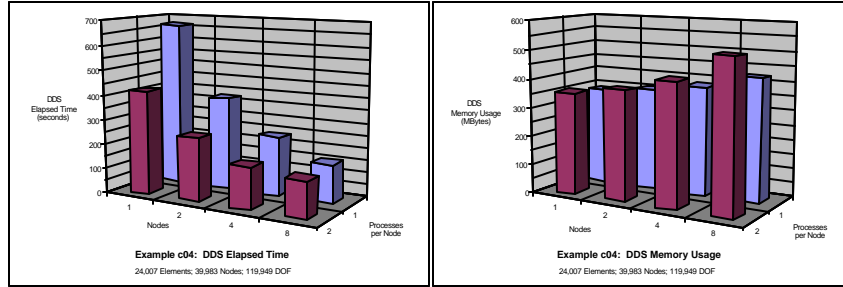
Example c03 - 12,538 Elements • 21,766 Nodes • 65,298 DOF

A number of trends are immediately apparent in the charts for examples c01 through c03:

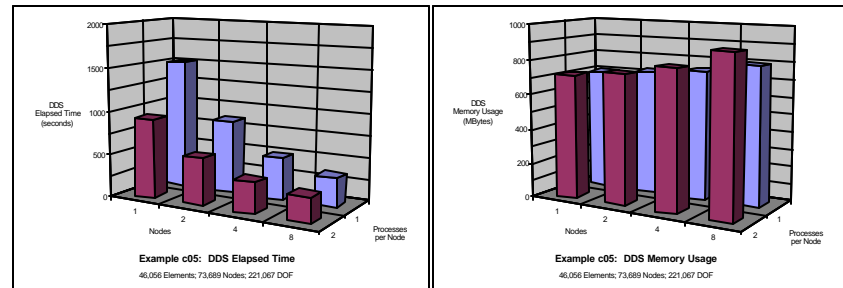
- Similar timings generally result from the use of equal process counts
- Single process per node timings are typically better than equivalent dual process runs
- Data replication levels are significant in this range of problem sizes
- Scalability is good, but not exceptional

Full machine tests (16 processes) consume about twice as much memory (cumulatively) as the corresponding single node runs. However, looking ahead, we see that this trend diminishes as the problem sizes increase, thus giving indication of an acceptable level of data replication - ultimately.

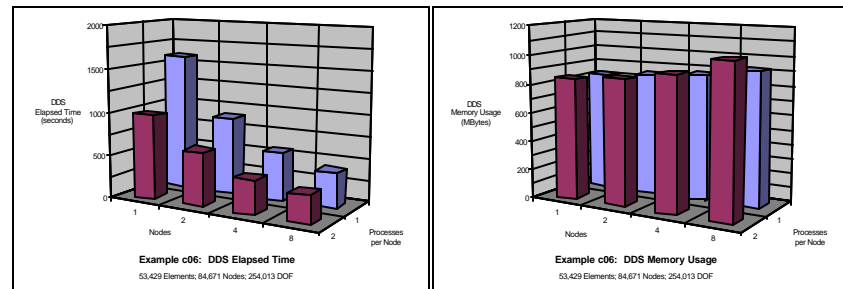
A couple of anomalies are apparent in the elapsed timing charts for examples c01 and c03 - the 8x1 and 2x1 cases, respectively. Neither is a limiting case, which suggests that load balancing problems may be more pronounced when node and subdomain counts are small.



Example c04 - 24,007 Elements • 39,983 Nodes • 119,949 DOF



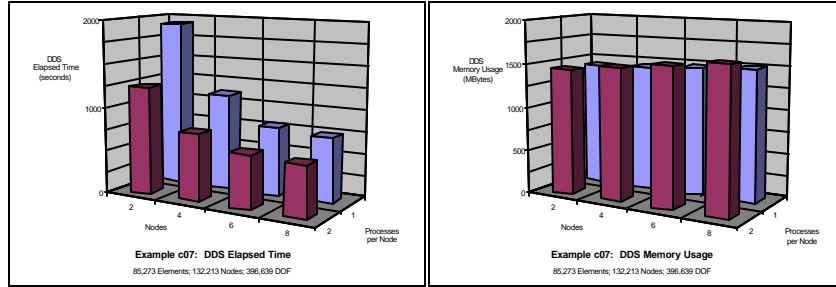
Example c05 - 46,056 Elements • 73,689 Nodes • 221,067 DOF



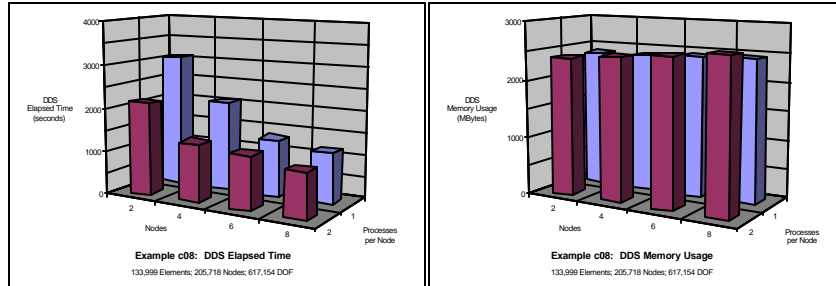
Example c06 - 53,429 Elements • 84,671 Nodes • 254,013 DOF

The second set of examples (c04 through c06) clearly exhibits better behavior. No significant irregularities were observed. The difference in elapsed times for equal numbers of processes approaches zero, but the set of single process per node timings remains slightly better in most cases. The proportion of replicated data subsides. Scaling remains solid, but unimpressive.

Examples c07 and c08 could not be run on a single node (1x1 and 1x2 cases) due to insufficient memory. However, sets of eight readings were maintained by the addition of 6- and 12-process configurations (6x1 and 6x2).

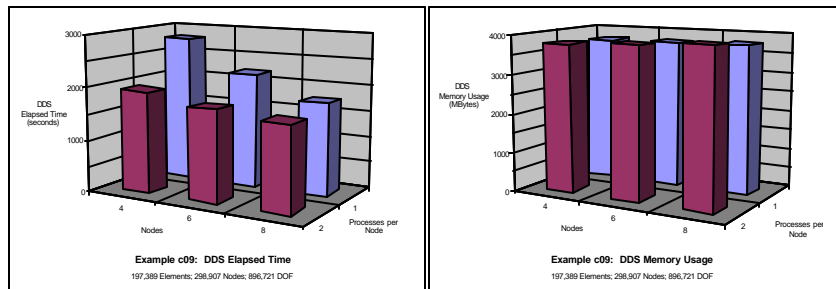


Example c07 - 85,273 Elements • 132,213 Nodes • 396,639 DOF

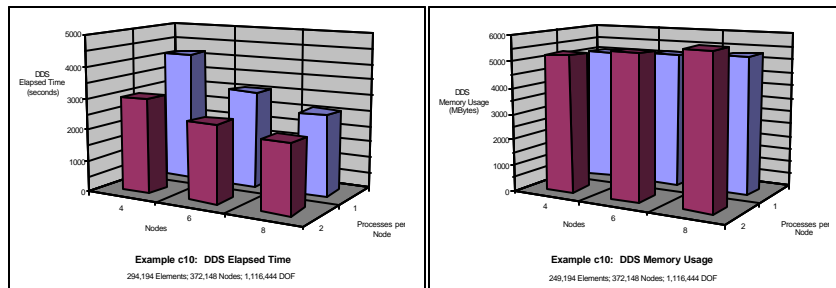


Example c08 - 133,999 Elements • 205,718 Nodes • 617,154 DOF

The largest examples we were able to run (c09 and c10) required a minimum of four node memories. For these cases, sets of only six readings are presented.



Example c09 - 197,389 Elements • 298,907 Nodes • 896,721 DOF

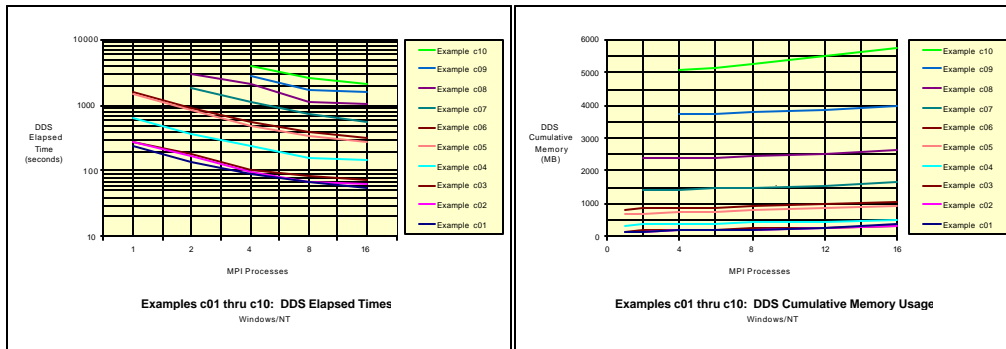


Example c10 - 294,194 Elements • 372,148 Nodes • 1,116,444 DOF

Single process-per-node timings maintained their slight advantage over the corresponding dual-process cases. Memory requirements rose dramatically for the largest test cases, while overheads remained modest

in relative terms. Attempts to resolve a case involving approximately 2M DOF consistently failed on a stack overflow reported by the compute node running MPI process 0.

Line graphs summarizing elapsed times and memory requirements across all cases with respect to partition sizing are presented in closing. Elapsed timings scale pretty well through eight processes. The fact that our Windows/NT test platform featured eight compute nodes is no coincidence. Obviously, the sharing of a single PCI bus and network card by multiple processors tends to have a negative effect on efficiency. Scalability beyond eight processes is poor.



Scalability of Windows/NT Results

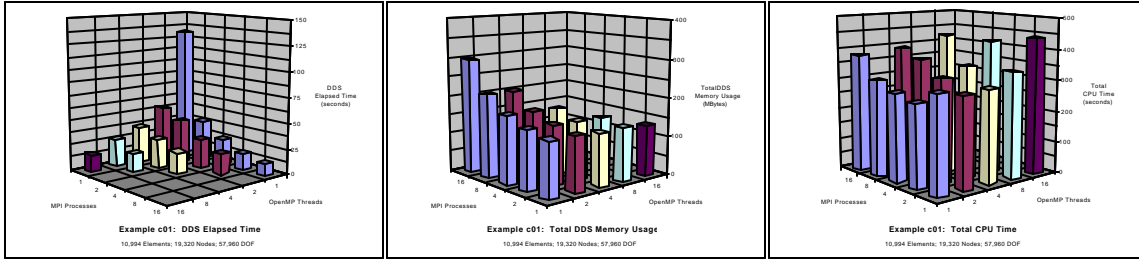
Memory usage remains almost flat throughout the entire range of problem sizes. Memory overheads are also nearly constant at around 250 Mbytes for all but the largest case (Example c10). It required more than 500 Mbytes. This particular result is significant because it occurs in proximity to the largest problem we were able to solve.

It is important to note that the modest slope of the memory usage curves gives clear indication of decreasing memory usage on a per-process basis, regardless of problem size. However, the node that runs MPI process 0 always consumes more than an equal share of the cumulative total and therefore, estimates based on an equal split are misleading. Tests performed by ANSYS suggest that the node running MPI process 0 requires an additional 10 to 20 Mbytes to accommodate MPI communication buffers used during the data distribution phase.

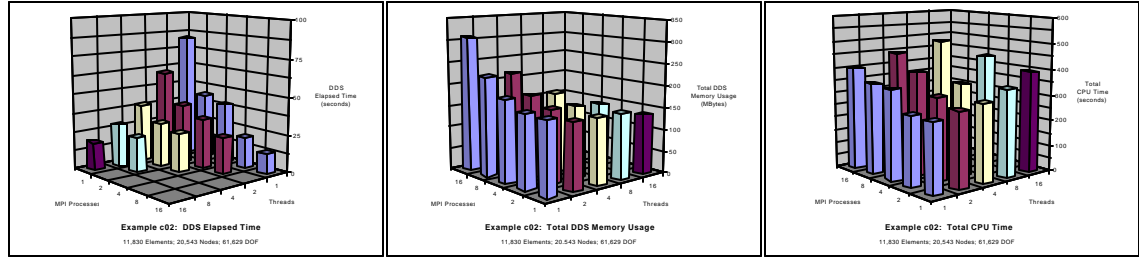
We may conclude that any Windows/NT cluster built to run the ANSYS DDS executable should be designed with the effects of non-uniform division of memory and network bandwidth in mind. The addition of processors without corresponding increases in the memory size and data transfer rates may not be warranted.

Origin 2000 Results

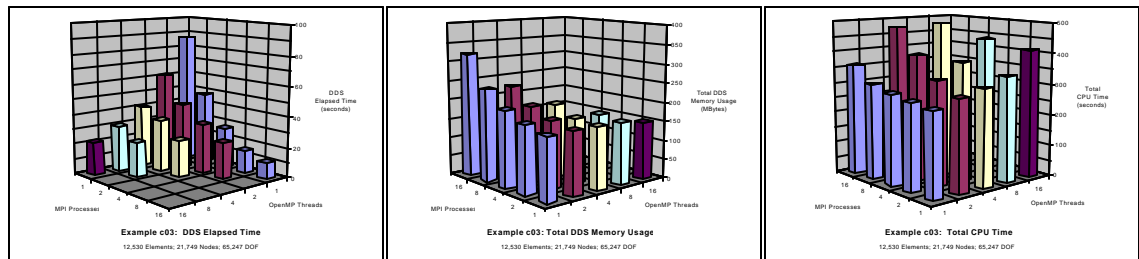
Each of the following chart groupings illustrates DDS elapsed times, cumulative memory usage, and overall CPU time consumed by the job.



Example c01 - 10,994 Elements • 19,320 Nodes • 57,960 DOF

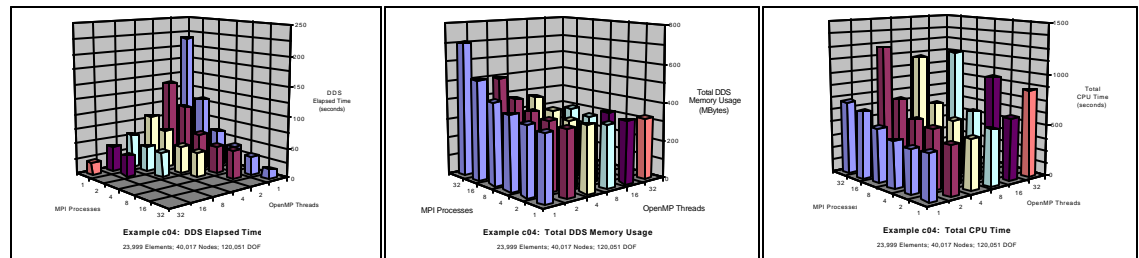


Example c02 - 11,830 Elements • 20,543 Nodes • 61,629 DOF

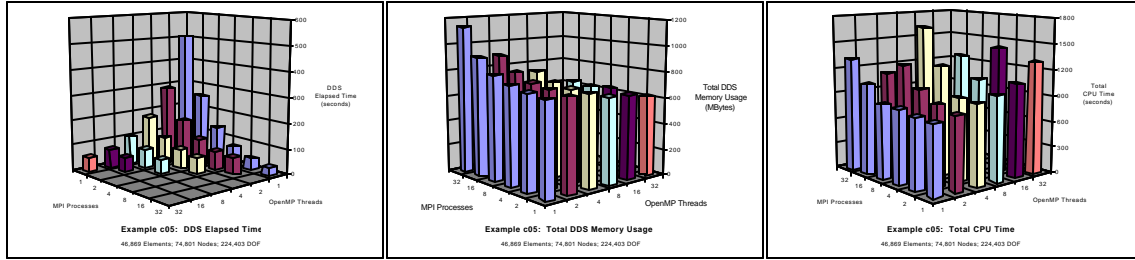


Example c03 - 12,530 Elements • 21,749 Nodes • 65,247 DOF

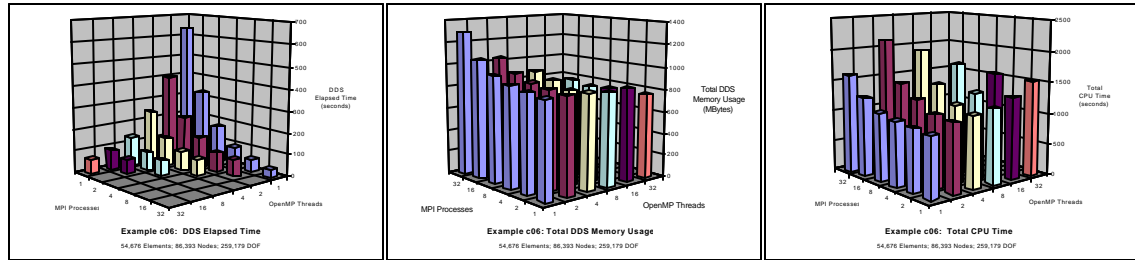
A new series of 32-way process and thread combinations has been introduced in the next set of charts.



Example c04 - 23,999 Elements • 40,017 Nodes • 120,051 DOF



Example c05 - 46,869 Elements • 74,801 Nodes • 224,403 DOF



Example c06 - 54,676 Elements • 86,393 Nodes • 259,179 DOF

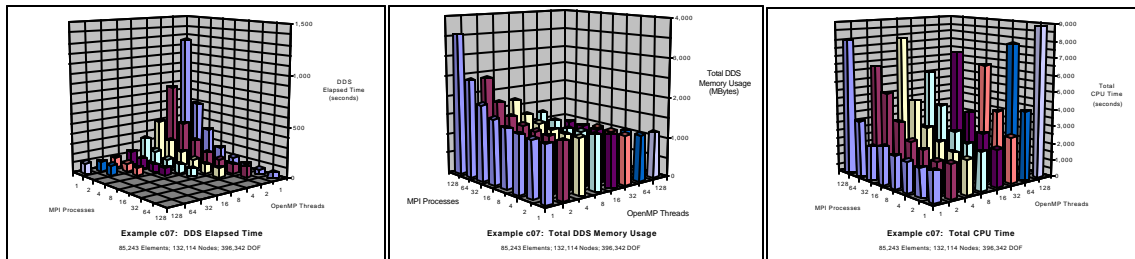
Three trends are clearly established by these smaller examples (50K to 250K DOF):

- Elapsed time is minimized by specifying of all MPI processes
- Total CPU time is minimized by specifying all MPI processes
- Memory requirements are minimized by specifying all OpenMP threads

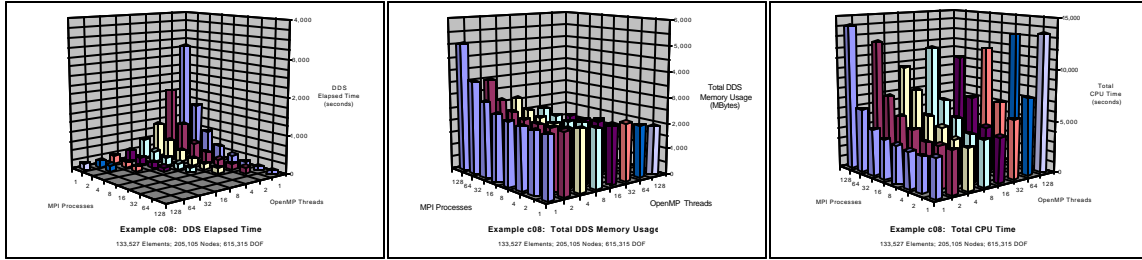
A few irregular results are apparent, but the overall behavior is very consistent. We believe that the above trends are likely valid for any smaller problem, and perhaps for any problem regardless of its size. In other words, resource utilizations for models represented by similar numbers of degrees of freedom will tend to track the same way. Solution times may vary.

The series involving two OpenMP threads per node didn't perform as well as we'd expected. Since Origin 2000 computers are constructed of dual-processor boards sharing a common memory space, we thought that dual-thread process configurations might perform optimally. Computational inefficiencies associated with the IRIX OpenMP model have been shown to be minimal.^{xii} Contention for shared memory locations may be occurring.

In the following examples, the range of partition sizes has been extended to include 64 and 128 process/thread combinations. This change was affected for essentially the same reason as before, e.g. to determine points at which the scaling of elapsed time goes into decline.

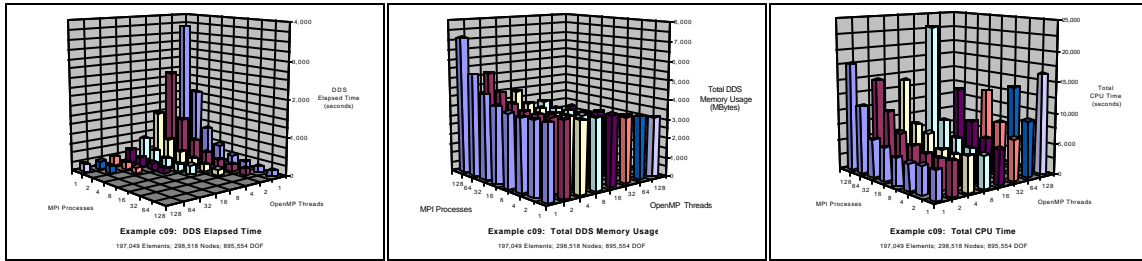


Example c07 - 85,243 Elements • 132,114 Nodes • 396,342 DOF

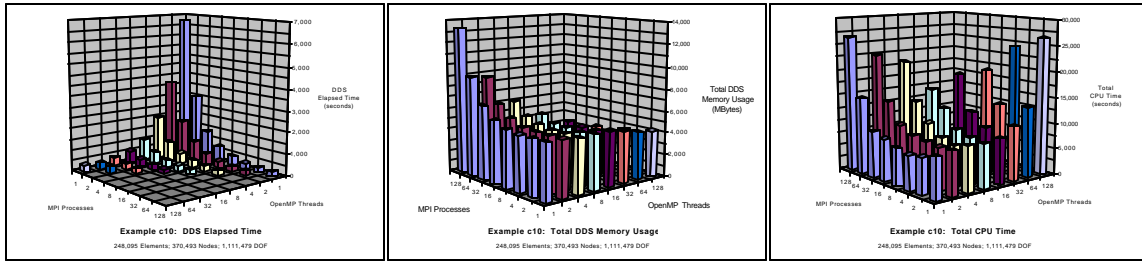


Example c08 - 133,527 Elements • 205,105 Nodes • 615,315 DOF

Trends established earlier are still evident. New series involving 64 and 128 process/thread combinations show incremental improvement for most cases, but memory consumption levels are excessive when few threads are used. Disproportionate spikes in total CPU time occur along both of these diagonals.



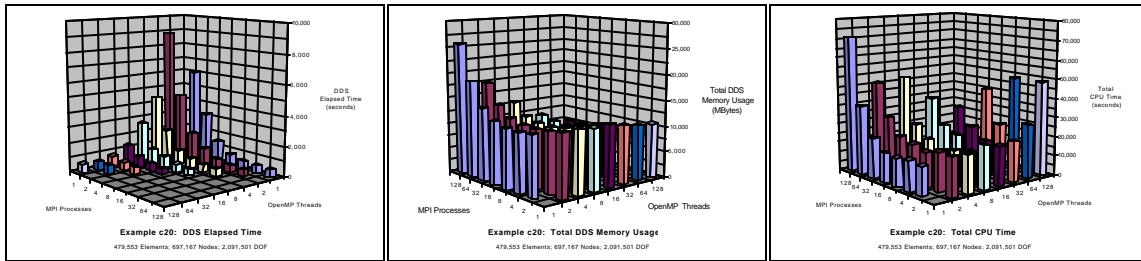
Example c09 - 197,049 Elements • 298,518 Nodes • 895,554 DOF



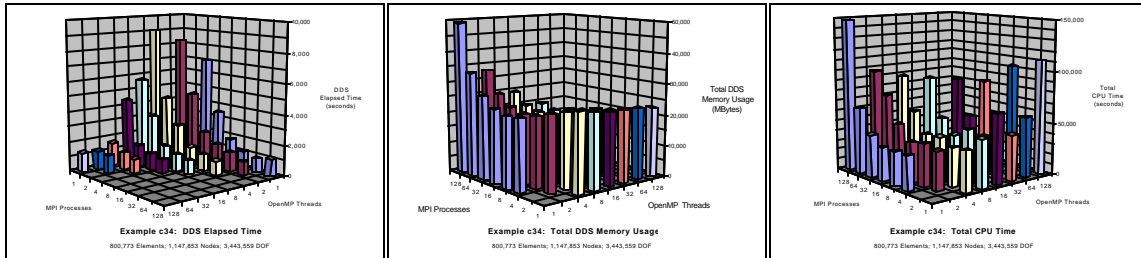
Example c10 - 248,095 Elements • 370,493 Nodes • 1,111,479 DOF

It is important note here that abnormal variations in DDS elapsed times and total CPU times were observed for virtually all of the 128-processor cases when executed on 128-processor machines, and so we re-ran all of these tests on a 256-processor machine. The implication is that similar variations might be expected to occur whenever a specified partition size equals the physical machine size.

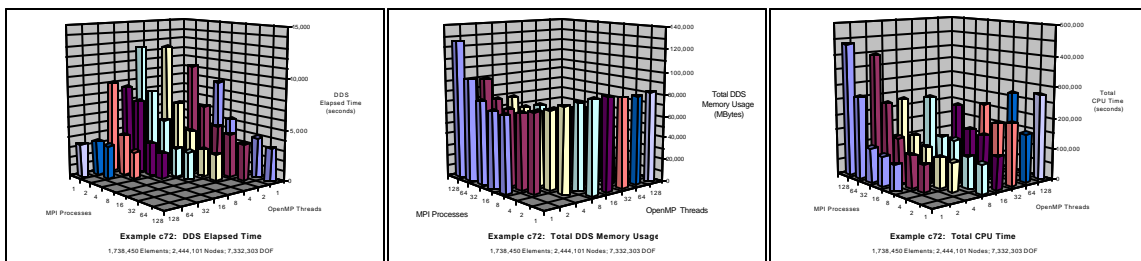
With serial solution times increasing rapidly, we elected to forego testing of some of the smallest partitions. Note the affect this has on the appearance of the final set of DDS elapsed time charts.



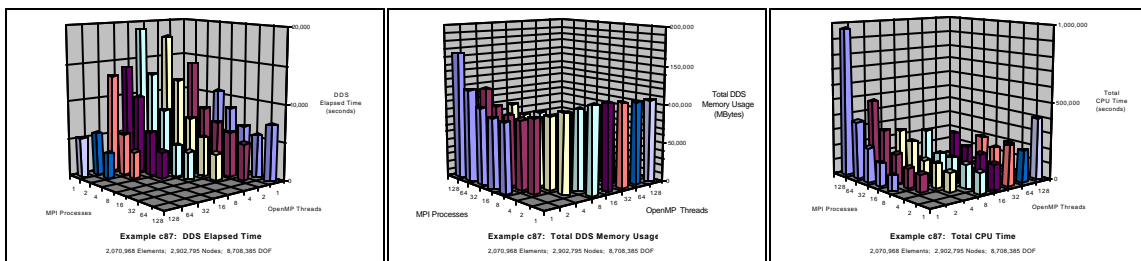
Example c20 - 479,553 Elements • 697,167 Nodes • 2,091,501 DOF



Example c34 - 800,773 Elements • 1,147,853 Nodes • 3,443,559 DOF



Example c72 - 1,738,450 Elements • 2,444,101 Nodes • 7,332,303 DOF



Example c87 - 2,070,968 Elements • 2,902,795 Nodes • 8,708,385 DOF

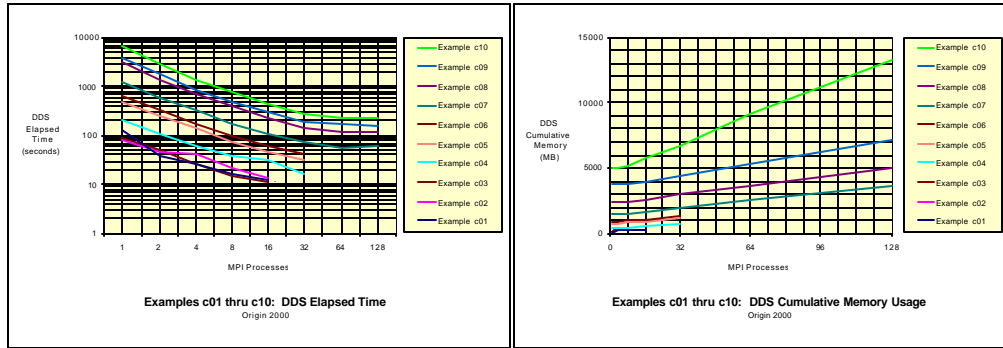
Results for the largest cases (c20 through c87) show a new trend towards optimal memory consumption for a mix of MPI processes and OpenMP threads. With a few notable exceptions, DDS elapsed and total CPU times are still minimized by using all MPI processes.

Additional guidelines for executing DDS analyses on Origin 2000 platforms include:

- Specified partition sizes should fit easily within the physical machine
- For larger problems, a good general purpose partition size is 32
- For smaller problems, all-MPI configurations should be favored
- To conserve memory, try eight MPI processes with four threads per process

Speedup values continue to rise through 128 processors, but total CPU times also show marked increases. This clearly indicates that for the largest partition sizes, improvements in turnaround times are being achieved inefficiently.

The following charts summarize DDS elapsed times and cumulative memory usages for examples c01 through c10. The given curves reflect all-MPI process measurements. Note that the scale of the x-axes (process counts) differs. The first chart uses a logarithmic scale; the second is linear.

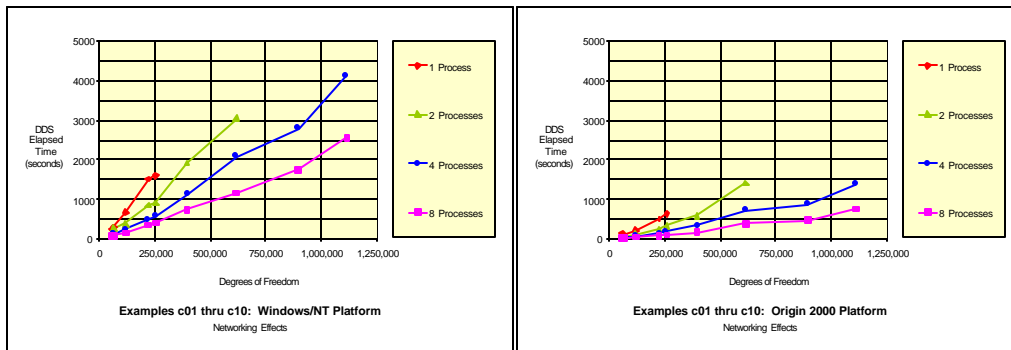


Scalability of Origin 2000 Results

The greatest fraction of the measured speedups is unmistakably realized between 2 and 32 processes regardless of problem size. The cost of data replication is generally modest in this range (~500 MB) thus confirming the recommendation to consider partition sizes that involve 32 processes.

Comparison of Window/NT and Origin 2000 Results

To approximate the effects of networking performance on our DDS timings, results for similar examples running as single and multiple process jobs were compared. All of the curves in the following line graphs reflect timings for a single process per node (Windows/NT) and a single OpenMP thread per MPI process (Origin 2000).



Scalability as a Function of Problem Size

The ratio of serial elapsed times for the Windows/NT and Origin 2000 systems remains nearly constant (approximately 3) throughout the noted range of problem sizes. When communications are present, this ratio increases only slightly (approximately 4), but the change is most noticeable in the curves for eight processes. Therefore, we might expect the difference to continue to grow for larger partitions as the Windows/NT cluster network becomes increasingly saturated.

We can continue this line of thought by characterizing the test platforms in terms of their non-uniform memory addressing (NUMA) ratios. The Windows/NT platform featured a 32-bit data path running at 66 MHz and standard 100 Mbit/s Ethernet cards, which evaluates to a factor of around 25 (66 MHz x 4 bytes/cycle ÷ 10 Mbyte/s). A similar estimate for the Origin 2000 architecture taken from SGI literature is about 2.5.

Implications of these statements are:

- The fraction of time spent on local computations scales at a factor of 3
- Overall timings where communications are present scale at a factor of 4
- The fraction of time spent on data communications scales at a factor of $25/2.5 = 10$

These values give indication that the fraction of time spent in data communications is about 14% of the total run time as illustrated by the following formulation of Amdahl's Law.^{xiii}

f_c = fraction of time spent in communication operations
 f_L = fraction of time spent on local memory operations = $1 - f_c$

$$3f_L + 10f_c = 4$$

$$3(1 - f_c) + 10f_c = 4$$

$$f_c = 0.1429$$

We know that this estimate holds for problems ranging from 60K to 1M degrees of freedom running on smaller partitions because it was derived from the experimental data. However, the Window/NT cluster network appeared to have been poised for a drop in performance as evidenced by increasing differentials in the timings for cases involving eight nodes.

Conclusions

Element reordering and I/O demands dominate turnaround times for large models. Since the reordering of elements for the DDS solver is unnecessary (maintained to support regression testing), it should have been removed from the 5.7 process stream. The cost becomes unreasonably high for larger problems due to poor cache utilization. Plans to eliminate this bottleneck should be considered first.

Specification of large workspace settings (- m and - db) eliminates virtual memory activities associated with the host program, leading to significant reductions in elapsed times. Although memory allocation is reportedly automated in an optimal fashion, this approach doesn't work anywhere near as well as the initial specification of a multiple gigabyte workspace. This feature should be reviewed.

Data communications files can be exceptionally large. Test cases required about 1 Kbyte per degree of freedom, or almost twice that of the corresponding restart file. Even under the best of circumstances, maintenance of multi-gigabyte data files is generally not a casual matter. Alternatives for a more efficient data distribution method should be considered.

Other conclusions that follow from our test data:

- Commodity clusters constructed of single bus, multi-CPU nodes may perform inefficiently when multiple processes per node are specified
- High performance SMP clusters demonstrate minimal memory requirements when running threads-only configurations, while elapsed and total CPU times are generally minimized by specifying all-MPI processes
- For medium to large problems, optimal resource utilization tends to occur in proximity to 32 process/thread combinations

Finally, a threads-enabled Windows 2000 product should be developed for multiprocessor PCs. The reduction in per-node memory requirements would be significant. Performance might also be enhanced if

local references are handled without generating messages. This specific change could potentially yield an extremely well balanced system capable of solving problems comprised of a few million degrees of freedom.

Acknowledgements

We wish to express our sincere thanks to Mark Straka, Robert Pennington, Qian Liu, and Louis Hoyenga of NCSA for their invaluable assistance in configuring the initial Origin 2000 and Windows/NT environments for our tests.

We would also like to thank Yong-Cheng Liu, Po-Shu Chen, Dave Conover, Lou Muccioli, and Lisa Fordanich of ANSYS, Inc. for provision of technical support.

Resources for this project were provided by NCSA under contract to ANSYS, Inc. Licensing for the MPI/Pro package was granted by MPI Software Technology, Inc.

References

ⁱ C. Farhat and F. X. Roux, "A Method of Finite Element Tearing and Interconnecting and its Parallel Solution Algorithm," *International Journal for Numerical Methods in Engineering*, Vol. 32, pp. 1205-1227 (1991)

ⁱⁱ C. Farhat, J. Mandel and F. X. Roux, "Optimal Convergence Properties of the FETI Domain Decomposition Method," *Computer Methods in Applied Mechanics and Engineering*, Vol. 115, pp. 367-388 (1994)

ⁱⁱⁱ Ibid

^{iv} <http://cas-www.colorado.edu/~charbel/Publications.html>

^v <http://www.psc.edu/~oneal/ansys/>

^{vi} <http://www-users.cs.umn.edu/~karypis/metis/>

^{vii} <http://www.mpi-softtech.com/>

^{viii} <http://www.sgi.com/software/mpt/>

^{ix} <http://www.pgroup.com/>

^x <http://www.platform.com/>

^{xi} <http://mauischeduler.sourceforge.net/>

^{xii} D. O'Neal, R. Luczak and M. White, "CAPTools Project Final Report: Evaluation and Application of the Computer Aided Parallelisation Tools," proceedings of the DoD HPCMP Users Group Conference, Albuquerque, NM (2000)

^{xiii} D. O'Neal and J. Urbanic, "On Microprocessors, Memory Hierarchies, and Amdahl's Law," proceedings of the DoD HPCMP Users Group Conference, Monterey, CA (1999)