Expanding the Use of Parallel Processing in a Commercial Structural Analysis Code

Eugene L. Poole, Cray Research, Inc., Eagan, Minnesota

ABSTRACT: *This paper describes new work with the ANSYS structural analysis finite element code which significantly expands the use of parallel processing on CRAY PVP (parallel/vector processor) machines. Many key applications programs on CRAY systems now have parallel solvers. The ANSYS program, a widely used general purpose Finite Element analysis code, has now expanded parallel processing to include parallel element computations and CFD analyses. The parallel processing model used by the ANSYS code for parallel element generation required the use of Cray macrotasking. The results presented will show that macrotasking and autotasking models work very well within a single code reducing time to solution, for single jobs as well as multiple runs on multi-user systems. The improvements in performance are demonstrated both for very large and smaller model nonlinear analyses. Examples will be given from real customer benchmarks on the CRAY J90 and C90 systems.*

Introduction

This paper describes new work with the ANSYS structural analysis finite element code which significantly expands the use of parallel processing on CRAY PVP machines. The new work demonstrates effective use of two different implementations of parallel processing within the same application and extends the use of parallel processing from very large model analyses to small and medium models used in many nonlinear analyses.

The new parallel processing capabilities in ANSYS Version 5.2 are motivated by the availability of multi-processor computer systems with shared memory architectures and similar parallel processing functionality across different hardware platforms. Shared memory architectures of CRAY PVP machines and several current high end workstation platforms provide a more convenient platform to adapt large existing codes for parallel processing. Common functionality on the various shared memory machines allows vendors to design a parallel processing model which can be implemented with few hardware specific changes required in the source code. This approach does not lead to scalable performance on MPP (Massively Parallel Processing) type architectures but is effective for small numbers of powerful processors and is most practical for use with existing codes where complete rewrites of the program are not currently considered practical.

The organization of this paper is as follows. First, a description of the use of parallel processing in the ANSYS code is given. Past work to exploit parallel processing capabilities on CRAY systems in both frontal solvers as well as I/O operations is summarized. New work which uses CRAY macrotasking calls to implement a new ANSYS parallel processing library is described. Design changes made by ANSYS, Inc. to the ANSYS code to make parallel processing of element computations possible are summarized and the implementation of the parallel processing library for CRAY systems is described. Second, results are presented from several representative ANSYS analyses. The results demonstrate improvements in time to solution for both large and small problems on both the CRAY C90 and J90 systems. The improvements in time to solution come from both increased use of parallel processing and improvements resulting from the use of the ANSYS PowerSolver and new CRAY optimizations for this solver in Version 5.2.

Description of ANSYS 5.2 Parallel Processing

The new parallel processing capabilities in ANSYS 5.2 add to the existing use of parallel processing in this general purpose finite element analysis code. This section briefly describes previous work to exploit parallel processing on CRAY systems in the ANSYS program. The previous use of parallel processing includes autotasking directives in the ANSYS frontal solvers and parallel I/O capabilities through the use of the CRAY EAG FFIO library. New parallel processing capabilities added in ANSYS Version 5.2 are described. The ANSYS parallel processing design for Version 5.2 is outlined in general terms and a description of the CRAY implementation of this library is given.

Parallel Frontal Solver and new Iterative PowerSolver

Previous parallel performance improvements in ANSYS were concentrated in computationally intensive frontal solvers [1] and [2]. The solvers were optimized for the CRAY PVP architecture without requiring extensive code changes. Sustained performance of as high as 6 Gflops were obtained on very large model analyses where the frontal solver dominated all other computations [3].

However, beginning with version 5.1 of ANSYS, a new iterative solver was added to the ANSYS program resulting in order of magnitude reductions in the computational cost of solving large linear systems. The iterative solver, now referred to as the ANSYS PowerSolver, is described in [4] with examples illustrating the reduction in computational cost compared to the frontal solver. The PowerSolver is a preconditioned conjugate gradient solver with an element specific preconditioning technique. The PowerSolver functions as a true "black-box" solver with no parameter adjustments required and is fully implemented in ANSYS, beginning with Version 5.1, as an alternative solver to the traditional frontal solver. The PowerSolver is optimized for CRAY architectures for the ANSYS 5.2 release with further improvements planned for the ANSYS 5.3 release including a parallel implementation and further vectorization improvements.

Parallel Processing of I/O

I/O performance, often a major bottleneck in reducing elapsed time to solution, was also reduced significantly beginning with ANSYS Version 5.0. Though not often considered as parallel processing, CRAY PVP systems use multiple levels of parallelism within the design of the I/O subsystem to deliver very high data transfer rates. The CRAY Applications EAG FFIO library, used extensively within many I/O intensive Applications codes, reduces elapsed time by using an intelligent read-ahead write-behind cache which runs concurrently with program execution see Refs. [1] and [2]). Large files, such as the ANSYS element files (emat, esav, osav and erot) and factored matrix files (tri) are written and read asynchronously, in parallel, to and from disks across multiple physical drive units. As a result, on multiple-user PVP systems very high rates of data transfer are achieved and real elapsed time speedups are possible for multiple CPU runs.

Parallel Processing Extended to Element Computations

The order of magnitude improvements in solver cost made possible by the new PowerSolver and increased use of nonlinear solution methods in ANSYS have shifted the focus of performance from solvers to element computations. Though element computations are inherently independent operations are thus well suited for parallel processing they present implementation challenges, particularly for large-scale existing codes.

In finite element codes elements are the basic building blocks. Physical models, often with complicated geometries, are divided up into simple triangular and/or square elements in 2-D models and tetrahedron and/or cubic solid elements in 3-D. The physical characteristics of the simple elements are represented through various mathematical models and the sum of these simple elements into a global model leads to an approximation of the solution of the large complicated physical system. In the ANSYS code over 100 different elements types are used as building blocks for a wide range of analysis types. The total number of elements typically used to represent a physical system ranges from a few thousand to hundreds of thousands.

Historically, in FEM (Finite Element Method) programs the cost of forming and evaluating elements was linear with the problem size while the cost of solving the global linear systems formed using the elements was quadratic with problem size. For larger and larger models the equation solution time grew to completely dominate the computational cost of analyses. However, with the use of new robust iterative solvers, such as the ANSYS PowerSolver, the cost of solving the global linear systems is now also linear with problem size so that the cost of element computations is much more dominant.

The cost of element computations, particularly on vector architectures, is further magnified by the fact that the element matrices are very small and the computation rates achieved on these computations are usually much lower than the rates achieved by the solvers. In addition, most commercial FEM codes have not been optimized to reduce the cost of element computations and they achieve a small fraction of peak performance on both vector and workstation platforms.

Due to the number of element routines in a typical general purpose FEM program and the generality of analysis capabilities desired the focus of design in these codes has been to concentrate on computing a single element at a time and the addition of this small unit of data to a very large database system. Often the control and execution of the element level computations exceeds the cost of the actual computations to form the element matrices or compute element results using the element matrices. Further work in this key performance area could result in order of magnitude reductions in computation cost for many programs. However, in most cases the changes required to improve the speed of element computations require redesign of a significant portion of the FEM program.

One method of reducing the cost of element computations is to use parallel processing. The element computations are largely independent and are natural choices for the use of multiple CPU resources. Even so, adding parallel processing to a large existing code requires careful design changes to the data structures and the logic in the code. In ANSYS version 5.0, released in March 1993, extensive design changes were made by ANSYS, Inc. to revise the database for the use of parallel processing. Version 5.1, released in October 1994, introduced the PowerSolver. Version 5.2, scheduled for release in October 1995, will feature parallel processing of element computations as well as additional parallel processing in the CFD analyses. The Applications division at CRAY has worked closely with ANSYS, Inc. to insure that the parallel processing model used for the element computations runs efficiently on CRAY systems while

preserving the existing parallel solvers and parallel I/O library from previous versions.

ANSYS Parallel Processing Considerations

The major considerations in parallelizing ANSYS element computations were the data used for elements and the actual computations of the elements. Globally, shared data is stored by the ANSYS program in common blocks separated by their function. In ANSYS a database file, stored in main memory with possible spillover to a disk file, stores all element data such as element types, node coordinates and material properties. The database is accessed through a single set of file buffers using a library of database access routines which copy specified data from the database to a set of buffers. In the parallel processing model the database is located in shared memory and the single set of file buffers are shared by all processors. This arrangement requires the use of locks, or critical regions, to insure that only one processor can access the shared data or buffers at a given time. The critical region is required for every database access in parallel regions for both read and write operations. While this approach is not scalable it required the least change to existing code. On CRAY systems no noticeable degradation in parallel performance from the critical regions has yet been observed.

In addition to database access, all shared data within parallel regions is stored in named common blocks or passed into element routines as arguments. Data which is local to each process is declared within the given routine. The parallel processing model used by ANSYS assumes that variables declared within a subroutine are local unless in a named common block. In this manner. no vendor specific variable declarations were required in the ANSYS routines to identify local and shared variables. This convention follows the CRAY multitasking models and required no CRAY-specific changes.

Finally, parallel computations in the element routines were isolated by calling driver routines in parallel from a common subroutine. 1 illustrates the general flow of computations in an FEM program, identifying three major areas where element routines are used. Element geometries are typically checked for errors in specified input or for excessive distortion. These element checks require no interprocessor communication other than that required to access the data for each element. Element formation typically involves extracting the element specific geometry and material information for each element from the global database and the forming one of more small element matrices through calls to various routines depending on the specific element type chosen.

For nonlinear analyses the global linear system results are used to compute element level results and the new results are used to update the element matrices or the loads applied, or both. If elements are recomputed at each iteration the element formation is repeated each time. If only loads are updated the same linear system is solved with new loads and the element results are recomputed. The iterative process continues until some convergence criteria is satisfied. For each element computation block in Figure 1 element data is extracted from the global database for each individual element and computations using that data are carried out with possible I/O operations at the end of the computations.

Figure 2 illustrates the general flow of computations for the element formation block in Figure 1. Even though many hundreds of routines are involved in computing the elements used in ANSYS only a few routines with calls to parallel processing library routines were required to execute the element computations in parallel. A general setup routine is called to initialize data structures and files used for element matrices. The setup routine calls a parallel driver routine which calls the element driver routine on multiple processors.

Each element driver routine executes identical code but obtains unique elements via critical regions where each processor obtains the next element. The beginning of the element driver routine is structured to allow only one processor at a time to obtain the next element number and required information for that element. Computations for that element then proceed by calling the appropriate element routines. When element computations are complete the element matrices are written to a single element matrix file, again in a critical region. The lock variable used for output is different from the database lock so different processors can be active in each critical region simultaneously. Though the computations required to form each element matrices are dominant this model of element computations requires very efficient hardware and software support for the critical regions. I/O bottlenecks on any hardware system would also reduce any potential benefit from parallel processing. Hardware specific changes to implement parallel processing were confined to a set of library routines provided by ANSYS to each vendor. The basic structure of the ANSYS parallel processing library is described next.

ANSYS Parallel Processing Library

Figure 3 lists the major components of the ANSYS parallel library. This library was designed in prototype form and modified with specific system calls for each hardware system. For example, the PPFRKn routines call the CRAY TSKSTART routine while PPLOCK calls the CRAY system routine LOCKON. The ANSYS parallel library defines all of the necessary functionality for parallel processing and confines hardware specific changes to these routines only. The parallel element computations are not loop-level parallel operations but rather are task-level operations. Each element operation typically involves database accesses and then computations using a set of nested subroutines to complete the task. All database accesses are locked so that the database file buffers are never used by more than one process at a time. Multiple lock variables are defined in the ANSYS parallel library and they are used according to function.

The ANSYS parallel library stores information required for the various parallel operations (such as the lock variables, the number of CPUS active, etc.) in a shared named common block. The information is read or modified through the use of parallel library routines, PPINQR and PPINFO, respectively. The

parallel library assumes a thread-like parallel processing environment and provides routines to resume and suspend threads (PPRESU and PPPARK) as well as barrier-like synchronization via PPSYNCI and PPSYNC. Individual processor identification is provided by the PPPROC routine. A call to this routine returns an integer from 0 to NPROC - 1, where NPROC is the number of processors used. The ANSYS program allows users to configure the number of processors used at run time as well as the level of parallel processing used. Users may run solvers only in parallel or elements only or both. The default configuration is to run with all parallel processing enabled. CRAY users may also choose to run ANSYS in parallel by just setting the environment variable NCPUS to the desired number of processors. Since this variable is set by default on many systems the ANSYS program will run in parallel by default.

CRAY Macrotasking Library for Version 5.2

The ANSYS parallel library functionally resembles a tasking form of parallelism. CRAY macrotasking calls for each function were easily implemented within the various routines. The ANSYS library also provided for thread based parallel models used on other parallel systems and much of this functionality was not required or used in the CRAY implementation. For example, the PPFRKn routines defined in the ANSYS parallel library are implemented using calls to the CRAY TSKSTART routine. If the number of processors, NPROC, is greater than one, then NPROC-1 calls to TSKSTART are executed, each calling the routine named in the call to PPFRK, with the master process calling the same routine. When NPROC-1 processes finish their work in the various element driver routines they return to the PPFRK routine where a call to TSKWAIT completes the PPFRK routine. The call to TSKWAIT automatically suspends the processes until the next PPFRK routine. The master process continues program execution. The ANSYS parallel library PPRESU and PPPARK routines are not needed on CRAY systems. The TSKWAIT routine also performs the function of synchronization, since program execution waits until all tasks complete, so the PPSYNCI and PPSYNC routines are also not used.

Timing considerations for the parallel processing computations provided a challenge within the ANSYS code. Though there are many ways to time user and system CPU and elapsed time within applications many of these standard routines do not work in parallel processing regions. For example, the ANSYS code uses the UNIX TIMES function to return elapsed time, user CPU time and system CPU time. A simple C routine is called from FORTRAN to compute these times from the C structures defined for the TIMES function. In parallel execution the times command does not return correct results for CPU times. Instead of using TIMES, the FORTRAN routine second is used to compute elapsed CPU time around each PPFRK call. The TIMEF function returns elapsed time for each PPRFK routine and the cumulative CPU and elapsed time for each parallel region are saved and printed at the end of program execution.

Results

The results presented in the following section come from runs with the pre-release version 5.2 of ANSYS. Most runs were made on non-dedicated systems and are given to demonstrate typical parallel processing performance in multi-user computing environments. They illustrate that significant elapsed time reductions occur using parallel processing even on multi-user systems. The major observed benefit from the new parallel processing features in ANSYS 5.2 is that a greater set of ANSYS problems now benefit from parallel processing. Many of these problems are very long running, I/O intensive nonlinear analyses where element computations dominate the compute time. Results are presented from runs on CRAY J90 systems for small and medium-sized jobs as well as some very large model examples from the CRAY C90.

CRAY C916 Large Model Examples

CRAY C90 results, demonstrated next, illustrate changes in ANSYS large model performance due to both parallel processing and the use of the iterative PowerSolver in place of the frontal solver. The parallel processing of elements effectively reduces time to solution as shown in the elapsed time comparisons with CPU time and the new solver technology further reduces solution time, even though the new solver runs at a substantially lower rate than the frontal solver.

Half-Million D.O.F. Example

Figure 4 illustrates the demands of the largest ANSYS models attempted on CRAY systems as well as the improvements in both solver and parallel processing performance from Version 5.1 to Version 5.2. The 574 D.O.F. model used in this analysis is one of the largest ANSYS examples attempted to date using a full nonlinear analysis. The analysis is designed to simulate cyclic thermal loading on a current design of a CPU chip used in high-end workstations. The loads on solder joints caused by wide temperature variations often causes failures and this analysis attempts to simulate this behavior using a detailed model of the chip components. Initially, the simulation was attempted using a CRAY C90 running ANSYS Version 5.0a. The ANSYS analysis was run for 24 hours on a dedicated machine using the parallel frontal solver with NCPUS set to 4. The analysis ran at a sustained rate of almost 1 Gflop, using 27 Gbytes of disk file storage and moving over 3/4 of a Terabyte of data. Nevertheless, only one half of the first thermal cycle was completed and completion of this analysis would have required nearly 2 weeks of dedicated time.

Recently, the same analysis was run using ANSYS 5.2 with the optimized PowerSolver and parallel element generation. This run was made on a non-dedicated system and was able to complete 18 cumulative iterations in the first thermal load cycle, 4 more than in the previous run, in half the elapsed time. Disk storage was reduced significantly as well as the data transferred due to the use of the PowerSolver. The table in Figure 4 compares solver time for a single iteration of this analysis. The PowerSolver requires 50 times fewer operations to obtain the

linear system solutions at each step. Though the frontal solver runs at peak rates on the CRAY PVP systems the PowerSolver is still significantly faster in execution time. Future optimizations are expected to double the computation rate for the PowerSolver and addition of parallel processing as well will reduce the time to solve the linear systems for each iteration from over 2 hours to just minutes. The parallel speedups in Figure 4 for the Version 5.2 run are entirely from parallel element computations and are comparable with the Version 5.0a runs where only the frontal solver was running in parallel.

Two Large Model Contrasting Examples

Figures 5 and 6 illustrate the changing nature of large model ANSYS runs resulting from new solver technology and parallel element generation. Figure 5 is a smaller 295k D.O.F. model analysis of the electronic chip described previously. This analysis was completed for 8 thermal load cycles on a CRAY C90 system using dedicated and multi-user runs over a one week period of time. The jobs accumulated 123 hours elapsed time with 88 hours of CPU time and sustained 255 Mflops. 8 Gbytes of disk storage were required and over 2 Terabytes were transferred to disk.

Recently, an engine cylinder head analysis was run for a single load step requiring 18 cumulative iterations. This model was run on a CRAY C90 system using ANSYS 5.2 and results were also obtained for this model run on a Sun Solaris workstation. The previous analysis using the frontal solver would probably not have been practical on a Solaris workstation, both for I/O considerations as well as elapsed time. The job would have required many weeks to complete. Now, with the PowerSolver this job can be solved on the workstation although the 600 Mbytes of main memory required for this analysis would completely dominate memory on most workstations. The sustained performance on both the Solaris and the C90 are well below their peak speeds. The computation rates reflect the much lower rates obtained for element computations. The CRAY performance for this job illustrates the performance of parallel element computations in ANSYS 5.2. The number of parallel routine calls was 42 for this analysis and over half of the CPU time of 3.7 hours was incurred in these parallel regions.

CRAY J90 Examples

The CRAY J90 results in this section illustrate ANSYS version 5.2 parallel processing performance on small models as well as a medium sized nonlinear analyses. A job throughput example illustrates parallel processing performance for an ANSYS 5.1 run of several jobs on an 8 CPU CRAY J90 system. All three examples illustrate parallel processing performance in multi-user environments on the CRAY J90 system.

Small Model Performance

Figure 7 illustrates ANSYS 5.2 performance for two small model examples. The first, a 7k D.O.F. Nonlinear Analysis run for the first of many load steps, runs at under 10 Mflops on a CRAY J90 system. The CPU time accumulated in 48 calls to parallel regions is nearly 2/3 of the total CPU time for the analysis. The parallel speedup in the parallel regions is over 3.6 (parallel CPU time divided by Parallel elapsed time) and the overall reduction in elapsed time is 1.8. Previously, small models runs such as this example, which run at very low Mflop rates on CRAY systems did not benefit at all from the parallel processing of the frontal solver. Now, with over 2/3 of the CPU time spent in parallel regions a significant elapsed time reduction is achieved. The total time for this small model analysis is many hours so the reduction in elapsed time due to parallel processing is a significant improvement over previous ANSYS performance. Models such as these also illustrate to need for continued performance improvements in the element processing computations.

A second example in Figure 7 shows parallel processing performance for a 31k D.O.K. eigenvalue analysis. Seven calls to parallel routines were executed in this run and the total elapsed time was less than the CPU time for the multi-user run even though the CPU time in the parallel regions was less than half of the total CPU time for the job.

Driveshaft Yoke Assembly Nonlinear Analysis Example

Figure 8 compares three runs of a complete nonlinear analysis of a driveshaft yoke assembly. The runs were all made on a lightly loaded CRAY J90 system using ANSYS 5.1 and 5.2 and also comparing the frontal solver in ANSYS 5.2 with the PowerSolver. The finite element model in this analysis was 39k D.O.F.s and the analysis used 15 separate load steps requiring a total of 189 cumulative iterations. Improvements in the PowerSolver performance on CRAY systems from ANSYS 5.1 to 5.2 are demonstrated by reduced CPU time for the 5.2. The major benefit in this analysis was from parallel processing since the elapsed time was nearly halved. The PowerSolver runs required a significant amount of I/O but only 1/3 of that required by the frontal solver. The ANSYS frontal solver achieved an impressive 281 Mflops of sustained performance on this job but it still slower than the PowerSolver in time to solution.

Parallel Processing Throughput Example

The final CRAY J90 example, Figure 9 demonstrates parallel processing performance for multiple job mix. The 3 jobs were run in a multi-user environment using the same ANSYS model. One job, JOB1, required 2 cumulative iterations while the remaining 2 were single load step static analyses. Each job ran with the environment variable NCPUS set to 3. The first job was started concurrently with the second on an 8 CPU CRAY J90 system which was also running 2 additional jobs. The third job started immediately after the second job finished. Each job finished in less elapsed time than CPU time and the total job mix finished in just over one hour. If the jobs were run in single CPU mode the first job would have required at least 1 hour and 14 minutes elapsed time, assuming no I/O wait time and no job swapping. These jobs all ran ANSYS Version 5.1 using parallel processing only for the frontal solver. Repeat runs of the individual jobs using Version 5.2 show further reduction in the

elapsed time when the element computations are also run in parallel.

This example demonstrates that the parallel processing used in the ANSYS code works not just for single large jobs on dedicated systems but is also effective on multi-user systems.

Summary

This paper has described new parallel processing improvements in the ANSYS program. The new improvements expand the use of parallel processing in ANSYS into small and medium sized models which require hundreds or even thousands of analysis iterations. Substantial reductions in elapsed time are demonstrated for customer supplied example from runs made on multi-user systems. The results demonstrate an improved capacity to solve a wide range of ANSYS analyses efficiently on CRAY systems. The efficient implementations of parallel processing for both autotasking loop-level parallelism and task-level macrotasking on CRAY systems coupled with the use of the EAG FFIO library to reduce I/O wait time deliver this performance within this important structural applications program.

References

[1] Eugene Poole, Dawson Deuermeyer, Doug Petesch, Omar Souka, Tom Hewitt, and Michael Heroux. The Impact of Cray Direct and Iterative Equation Solvers in Industrial Applications Codes. In *1992 Fall Proceedings Cray User Group*, pages 49--62, CUG Secretary, 1111 Stewart Avenue, D09-GHQ, Bethpage, NY 11714, September 1992.

[2] Eugene Poole, John Bauer, Troy Stratton, and Tom Weidner. Large-Scale Nonlinear Structural Analysis Simulation on CRAY Parallel/Vector Supercomputers. *Computing Systems in Engineering*, 4(4):405-414, 1993.

[3] Eugene Poole and John Bauer. Achieving GFlop Performance for ANSYS Analyses on the CRAY C916 Supercomputer. In *Proceedings of 1994 ANSYS Conference and Exhibition*, pages 13.3-13.12, ANSYS, Inc., P.O. Box 65, Houston, PA, May 1994.

[4] Eugene Poole, Mike Heroux, Pravin Vaidya, and Anil Joshi. Performance of Iterative Methods in ANSYS on CRAY Parallel/Vector Supercomputers. To appear in *Computing Systems in Engineering*, 1995.

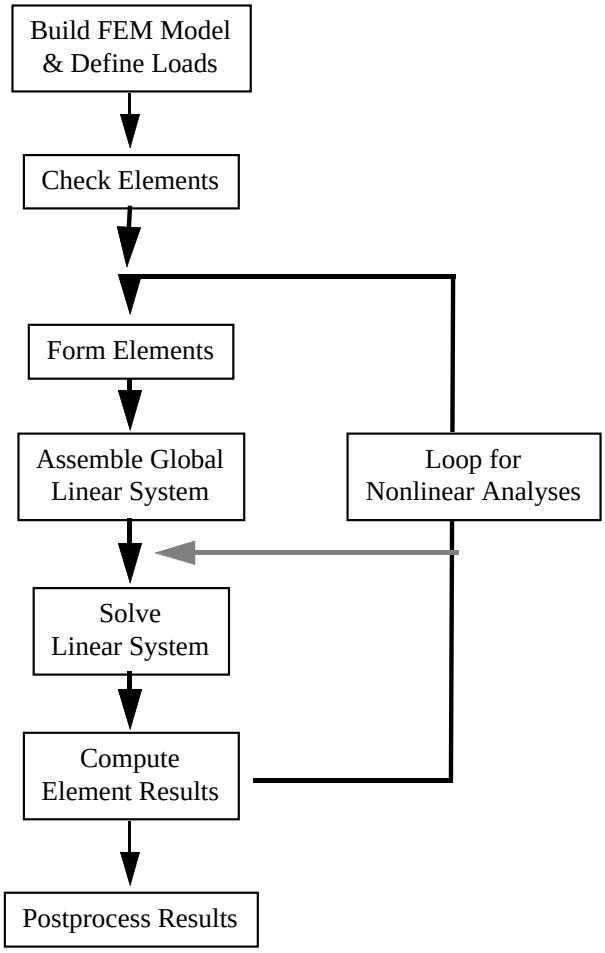

Figure 1: Finite Element Method Analysis Flowchart.

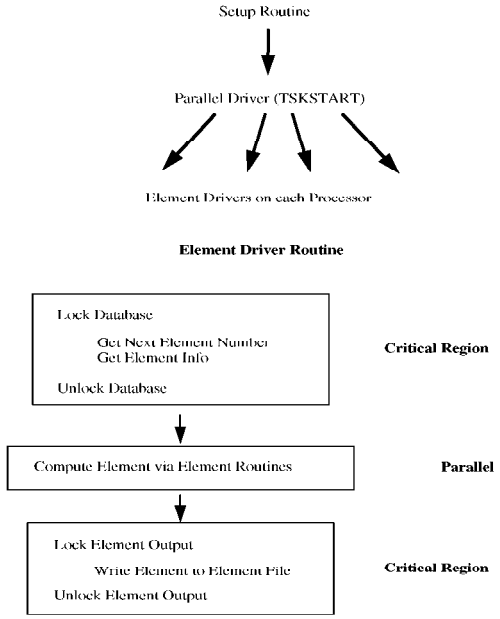

Figure 2: Parallel Element Generation Flowchart.

574,966 D.O.F. Model of Electronic Components
120,404 Solid 45 8-Node Elements
35,872 Viscoplastic Solid 8-Node Elements

CRAY C98
24 Hour Dedicated
Frontal Solver 5.1
4 CPUS

23.6 Hours Elapsed Time, 41 Hours CPU Time
14 Cumulative Iterations Completed
27 Gbytes Disk Storage
792 Gbytes Data Transferred to Disk

CRAY C98
Multi-User Run
PowerSolver 5.2
4 CPUS

14.3 Hours Elapsed Time, 24 Hours CPU Time
18 Cumulative Iterations Completed
7 Gbytes Disk Storage
208 Gbytes Data Transferred to Disk

CRAY C90 Single CPU Time Comparison

Solver	Work (Billions)	Rate (MFlops)	Time (CPU sec.)	Iter
Frontal WF:5230 *CRI optimized*	6500	859	7569	
PowerSolver *Version 5.1*	130	65	2000	1050
PowerSolver *Version 5.2*	130	127	1026	1050

Figure 4: 574K D.O.F. Large Model Nonlinear Analysis Example.

295,433 D.O.F. Model of Electronic Chip
62,870 Solid 45 8-Node Elements
15,200 Viscoplastic Solid 8-Node Elements

CRAY C90 Dedicated and Multi-User Runs Using 4 CPUs
118 Cumulative Iterations Completed, 8 Load Steps
ANSYS 5.0a Run Using Frontal Solver

123 Hours Elapsed Time, 88 Hours CPU Time
255 Mflops Sustained Performance
8 Gbytes Disk Storage
2 Terabytes Data Transferred to Disk

Figure 5: Large Model Nonlinear Analysis Using Frontal Solver

ANSYS ROUTINES | **CRAY SYSTEM ROUTINES**

PPINIT( )
PPRESU( )*
PPFRKn(routine,arg1,...argn) — *TSKSTART*
PPSYNCI*
PPPROC( ) — *TSKVALUE*
PPNEXT( )
PPLOCK(ilock) — *LOCKON*
PPUNLOCK(ilock) — *LOCKOFF*
PPSYNC*
PPINFO(key,info)
PPINQR(key)
PPPARK( )*
PPFINI

* Not Needed for CRAY Implementation

Figure 3: ANSYS Parallel Library Organization.

262,563 D.O.F. Model of Engine Cylinder Head
54,544 SOLID92 Tetrahedron Elements
1,027 LINK8 Elements

7 Gbytes Data Transferred
1 Gybte File Space Used
600 Mbytes Main Memory Used

ANSYS 5.1
Solaris Workstation

19.6 Hours Elapsed Time, 15.1 Hours CPU Time
5.3 MFlops Sustained Performance

ANSYS 5.2
CRAY C90
Multi-User Run
NCPUS=4

2.7 Hours Elapsed Time, 3.7 Hours CPU Time
35 Mflops Sustained Performance
42 Parallel Routine Calls
7503 CPU Sec. 3270 Wall Sec.

Figure 6: Large Model Nonlinear Analysis Using PowerSolver.

ANSYS Version 5.2

7,424 D.O.F. Model
1,770 Beam Elements
691 Shell63 Elements

CRAY J98 Multi-User Run
NCPUS=4
1st Load Step Only - 9 Cumulative Iterations

Total Job CPU Seconds
Total Elapsed Seconds

Parallel Regions Called
Parallel CPU Seconds
Parallel Elapsed Seconds

Small Model, Nonlinear Analysis Example

ANSYS Version 5.2

31,641 D.O.F. Model
9,270 Plane 42 Elements
515 Solid45 Elements

CRAY J98 Multi-User Run
NCPUS=4

1,167 Total Job CPU seconds
836 Total Elapsed seconds

7 Parallel Regions Called
479 Parallel CPU Seconds
124 Parallel Elapsed Seconds

Small Model, Eigenvalue Analysis Example

Figure 7: Small Model CRAY J90 Examples.

39,038 D.O.F.s, 8,836 Tetrahedron Solid Elements

15 Load Steps, 189 Cumulative Iterations

CRAY J90 CPU Time Comparison

Solver, Version	CPU (Hrs.)	Wall (Hrs.)	Rate (MFlops)	Gbytes Rd/Write	I/O Wait (sec.)
PowerSolver,5.1 *NCPUS=1*	18.7	19.2	7.6	95	1230
PowerSolver,5.2 *NCPUS=4*	18.1	9.9	13.7	104	1044
Frontal Solver,5.2 *NCPUS=4*	44.1	16.5	281	323	3442

Figure 8: CRAY J90 Driveshaft Yoke Assembly Analysis

68,937 D.O.F. Model
18,303 Solid45 Elements

CRAY J98 8 Processor System
NCPUS=4 for each job
3 Analyses Using Same Model

ANSYS Version 5.1

Job 1		Job 2		Job 3	
4,492	CPU Secs.	2,078	CPU Secs.	2,106	CPU Secs.
3,697	Wall Secs.	1,574	Wall Secs.	1,804	Wall Secs.

Total I/O Wait Time < 7 Minutes
Total Data to/from disk 12.6 Gbytes
All Jobs Completed in 3,697 Seconds