NX Nastran 10

PARALLEL PROCESSING GUIDE

Proprietary & Restricted Rights Notice

© 2014 Siemens Product Lifecycle Management Software Inc. All Rights Reserved. This software and related documentation are proprietary to Siemens Product Lifecycle Management Software Inc.
NASTRAN is a registered trademark of the National Aeronautics and Space Administration. NX Nastran is an enhanced proprietary version developed and maintained by Siemens Product Lifecycle Management Software Inc.
MSC is a registered trademark of MSC.Software Corporation. MSC.Nastran and MSC.Patran are trademarks of MSC.Software Corporation.
All other trademarks are the property of their respective owners.

TAUCS Copyright and License
TAUCS Version 2.0, November 29, 2001. Copyright (c) 2001, 2002, 2003 by Sivan Toledo, Tel-Aviv University, stoledo@tau.ac.il. All Rights Reserved.
TAUCS License:
Your use or distribution of TAUCS or any derivative code implies that you agree to this License.
THIS MATERIAL IS PROVIDED AS IS, WITH ABSOLUTELY NO WARRANTY EXPRESSED OR IMPLIED. ANY USE IS AT YOUR OWN RISK.
Permission is hereby granted to use or copy this program, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any derivative code must cite the Copyright, this License, the Availability note, and "Used by permission." If this code or any derivative code is accessible from within MATLAB, then typing "help taucs" must cite the Copyright, and "type taucs" must also cite this License and the Availability note. Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. This software is provided to you free of charge.
Availability (TAUCS)
As of version 2.1, we distribute the code in 4 formats: zip and tarred-gzipped (tgz), with or without binaries for external libraries. The bundled external libraries should allow you to build the test programs on Linux, Windows, and MacOS X without installing additional software. We recommend that you download the full distributions, and then perhaps replace the bundled libraries by higher performance ones (e.g., with a BLAS library that is specifically optimized for your machine). If you want to conserve bandwidth and you want to install the required libraries yourself, download the lean distributions. The zip and tgz files are identical, except that on Linux, Unix, and MacOS, unpacking the tgz file ensures that the configure script is marked as executable (unpack with tar xvpf), otherwise you will have to change its permissions manually.

Table of Contents

Preface - About this Book

NX NASTRAN is a general-purpose finite element program which solves a wide variety of engineering problems. The NX NASTRAN Parallel Processing Guide is intended to help you choose among the different parallel processing and computational methods, and ultimately increase the performance of analysis by reducing CPU time, memory and disk space requirements.

This main material in this book covers the parallel processing methods for the linear static, normal modes, direct frequency response, modal frequency response, modal transient response, and design optimization.

This book is composed of 7 chapters:

1. Introduction and fundamentals
2. Running parallel NX Nastran solutions
3. DMP computational methods for linear static analysis
4. DMP computational methods for normal modes analysis
5. DMP computational methods for response analysis and optimization
6. Performance study
7. Installation and Configuration of Distributed Memory Parallel (DMP)

To effectively use this book, it is important for you to be familiar with the basic structure of NX Nastran. For more information about the mathematical foundation, refer to *NX Nastran Numerical Methods User's Guide*.

This book will continue to be revised to reflect future enhancements to NX Nastran parallel processing methods. Changes and additions are encouraged, and can be communicate through Siemens PLM technical support http://support.ugs.com/.

Chapter 1 – Introduction and Fundamentals

1.1 Parallel Computing in NX Nastran

In many applications today, the volume of analysis is going up, the physics being modeled is increasingly complex, and the time allocated to development is under increasing pressure. Although the recent significantly advancement in computer environment, such as higher speed processors and larger memory, has eased the investigation into increasingly complex problems, there are still limitations on computational performance in serial computation even with optimized processes and data flows.

Parallel computing is an approach that uses multiple computers, processors or cores working together on a common task. Each processor works on a section of the problem and exchanges information (data in local memory) with other processors.

With parallel computing, NX Nastran provides simultaneous use of multiple processors on one or more machines to decrease solution times. The benefits of parallel processing include:

- Reduced solution time on large problems
- Reduced hardware requirements by utilizing smaller and less expensive machines
- Decreased turnaround time for each solution; allowing you to analyze more designs under more conditions in a shorter amount of time

The following are basic parallelism concepts in NX Nastran.

Types of parallelism. There are basically two types – data parallel and task parallel. In data parallel, each processor performs the same task on different data. In task parallel, each processor performs a different task. In NX Nastran, as in most real applications, parallelism falls somewhere on the continuum between these data and task parallelism types and involve more than one type of problem mapping.

Parallel computer architectures. The control mechanism of parallel computing in NX Natran is based on Multiple Instruction, Multiple Data (MIMD) which refers to a parallel execution model in which each processor is essentially acting independently. The processors work on their own data with their own instructions. Tasks executed by different processors can start or finish at different times.

Parallel programming models. NX Nastran supports shared memory (or threads) and message passing. Parallel programming models exits as an abstraction above hardware and memory architectures. In hybrid model any two or more parallel programming model are combined.

Message passing is a widely used communication model in programming parallel processor machines for DMP processing. You can access the communication hardware only through the message passing library. Message passing libraries like Message Passing Interface (MPI) or Parallel Virtual Machine (PVM) send and receive data from a program running on one node to a program running on another node. In particular, MPI is a standard message passing library that has been efficiently implemented on a variety of platforms.

MPI is portable, efficient, expressive, and provides thread safety. In addition,

- Point-to-point communications handle data transmission between any two processors in a communicator.
- Collective communications handle simultaneous communication between all processors in a communicator.

NX Nastran offers the ability to run certain solution sequences in parallel using a Message Passing Interface (MPI), an industry-wide standard library for C and Fortran message-passing programs. MPI programs can be run on SMP computers, DMP computers, and a cluster of computers supported by the MPI package. In most cases, NX Nastran uses the hardware vendor's MPI implementation. While this usually results in the highest performance levels, it also limits a DMP job to computers supported by the vendor's MPI package.

Further information on the MPI standard is available online at the MPI forum website: http://www.mpi-forum.org.

1.2 Keywords in NX Nastran Parallel Computing

The following keywords are used in parallel NX Nastran.

Execution keywords: dmp, hosts, slaveout, parallel, numseg, nclust, nrec, dstat.

Performance keywords: memory, scratch, sdirectory, sscr, buffsize, rdscale.

In the appendix, gpart is described as an execution keyword, and it is typically not required. More keywords are described in the *NX Nastran Installation and Operation Guide*.

1.3 Getting Started

The following steps give you a basic idea how to use parallel processing for the desired analysis in NX Nastran.

- **Step 1. Know your machine architectures.** SMP or DMP machine (see section 1.4). Memory size and etc.
- **Step 2. Choose parallelism scheme.** SMP or DMP scheme (see section 1.5). If SMP only, refer to running SMP section (2.2). Most solution sequences support SMP, as well as the RDMODES method (section 4.4). If DMP, refer to running DMP section (2.3). In many cases, a combination of SMP and DMP is possible.
- **Step 3. Select and run an appropriate method for the desired analysis.** Refer to the section “computational method at a glance” (section 2.1)
- **Step 4. Tune the performance.** Best performance may depend on the machine architecture. Refer to sections (2.4.4) and (4.5) for general guidelines.

1.4 Parallel machine architectures

Hardware can be divided roughly into two categories.

Shared Memory Parallel (SMP) Machines. A SMP machine is defined as a single machine with multiple processors that share a common memory and I/O as illustrated in Figure 1.1. Multiple processors can operate independently, but changes in a memory location affected by one processor are visible to all other processors. The primary disadvantage is the lack of scalability between memory and processors.

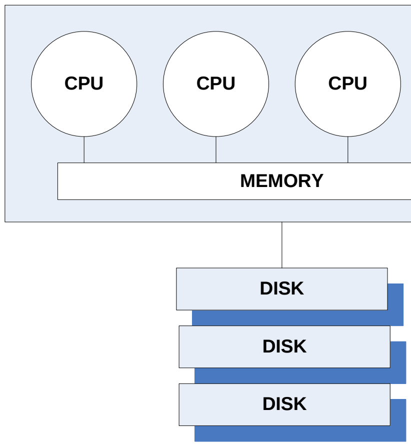


Figure 1.1 SMP Machine architecture

Distributed Memory Parallel (DMP) Machines. A Distributed Memory Parallel (DMP) machine uses multiple machines or clusters with one or more processors communicating over a network, or multiprocessors with multiple I/O channels. Figure 1.2 shows a typical architecture of a DMP machine. Each machine has its own memory and one or more disks. In DMP, data is private to each node and it is necessary to exchange data across different nodes. Therefore, the programmer must decide *which* data is to be sent/received to/from *which* node. The main advantage is that memory (and local disk) is scalable with multiple processors.

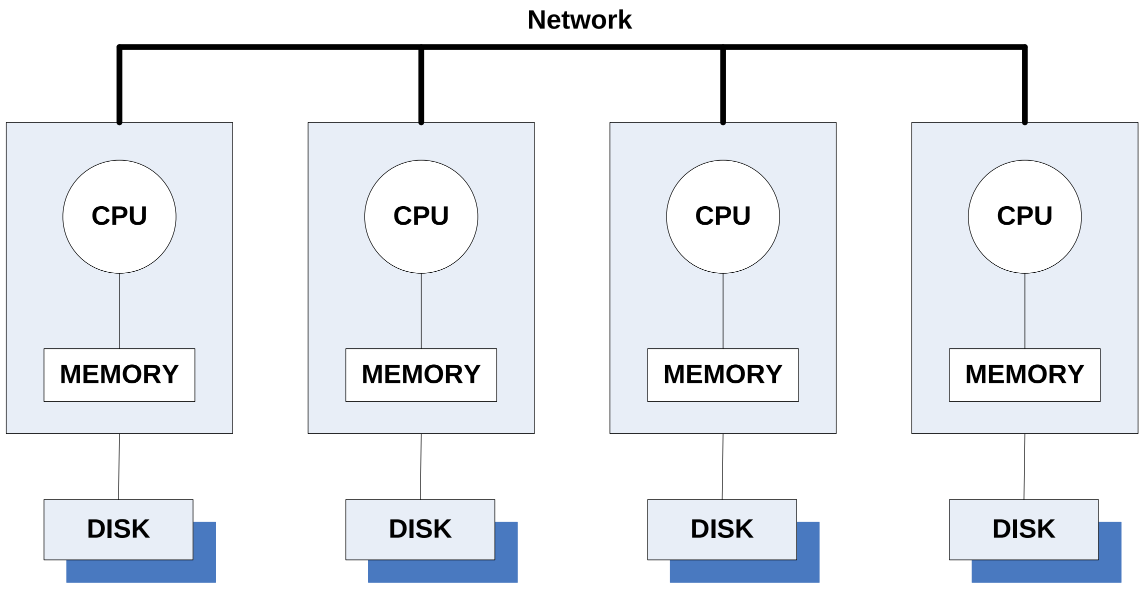


Figure 1.2 DMP Machine architecture

1.5 Parallelism in NX Nastran

NX Nastran supports both shared memory parallel (SMP) and distributed memory parallel (DMP) processing.

In NX Nastran, SMP is used only for lower level operations such as matrix decomposition and matrix multiplication for all solution sequences. Therefore, as long as suitable hardware is available, all solutions can utilize SMP processing.

The DMP is based on domain decomposition on geometry domain or frequency domain, or load domain. DMP methods achieve their solution speed by dividing the FE model into smaller pieces to be solved simultaneously. This division is performed with respect to geometry or frequency range individually or both at the same time. Although each processor is working on its own partition of the geometry or frequency range, it communicates with the others to share information. Once the solution is complete, the results are merged, creating a single result file.

Most of the discussions and examples in this book focus on the DMP computational methods and solution methods.

The differences between DMP and SMP are listed in the table below.

Feature	DMP	SMP
Hardware environment	High performance workstation clusters	Shared memory multi-processor workstations
Parallelism level	Partition finite element model	Subdivided matrix and vector operations
Software mechanism	Message Passing Interface (MPI)	OpenMP API(or pthreads)

1.6 Parallel Solution Methods in NX Nastran

NX Nastran offers several solution sequences in parallel for both static and dynamic analyses. As mentioned in previous section, two parallelisms (SMP and DMP) are available.

In SMP, it parallelizes the computing for decomposition and matrix multiplications. Since every solution sequence involves at least matrix multiplications, SMP can be activated in all solution sequences for all analyses as long as the hardware supports SMP. An SMP RDMODES run is available for SOL 103 and SOL 111 which can significantly reduce regular run time if an approximate solution is acceptable. See section 2.1 on how to activate SMP.

In contrast to SMP, which focuses on parallelizing computational modules (such DMCP, MPYAD), DMP provides parallelism on the algorithm level, which can provide greater speedup. They can be categorized into the following three methods:

- Domain Static Analysis (DSTAT) method (see chapter 3)
- Domain Normal Modes Analysis (DMODES) method (see chapter 4)
- Domain Frequency Response Analysis (DFREQR) method (see chapter 5)

They are for linear static analysis, normal modes analysis, and frequency response analysis, respectively. Modal frequency and transient responses require computing modes for modal space. Therefore, DMODES can be applied during mode computation. In design optimization, if it involves computing modes, DMODES also can be activated. In summary, DMP computational methods support the following solution methods.

●	SOL 101	Linear statics (see chapter 3)
●	SOL 103	Normal modes (see chapter 4)
●	SOL 105	Buckling (see chapter 4)
●	SOL 108	Direct frequency response (see chapter 5)
●	SOL 111	Modal frequency response (see chapter 5)
●	SOL 112	Modal transient response (see chapter 5)
●	SOL 200	Design optimization (see chapter 5)

The DMP computational methods will be discussed in chapters 3-5.

1.7 Expectation from NX Nastran Parallel Solutions

An NX Nastran parallel solution should able to solve a large problem with significantly less run time compared to a standard serial (single processor) run. Note that (as in most parallel codes), speedup will be less than the number of processors, due to MPI startup cost, communication overhead, and inherent limitations in the parallelizability of the algorithms used. Furthermore, if the computation can already be done on a single processor in minimal runtime, there will not be much opportunity for improvement, especially for small problems (less than 10,000 DOFs).

Chapter 2 – Running Parallel Solutions

An NX Nastran parallel job can be selected by the Nastran command with either ***keywords*** or ***system cells***. The Nastran command permits the keywords, **dmp** and **parallel**, to request DMP and SMP runs, respectively. Alternatively, system cells **231** and **107** can also be used for DMP and SMP runs.

It is strongly recommended to use ***keywords*** to request a parallel job. The following table provides keywords and their descriptions as well as the system cells that can be used for DMP and SMP runs.

	Keywords	**Description**	**System Cell**
DMP	dmparallel (or dmp)	Default = 0 (deselect DMP processing) Specifies the number of tasks for a DMP analysis 1-256 processors are available in a DMP job	231
SMP	parallel (or smp)	Default =0 (deselect SMP processing) Specifies the maximum number of CPUs selected for SMP processing 1-1023 processors are available in a SMP job.	107

2.1 Computational Methods at a Glance

The SMP computational methods are

Analysis	**Comp. Method**	**Submittal Command**	**Suggested Model Type**
All Analysis: **SOL XYZ**	**None**	parallel=*p*	All models

One of the computational methods that is based on the domain decomposition but can run in serial fashion (not DMP) is RDMODES, as an alternative to ACMS. It can reduce the runtime significantly even in single processor, compared to a conventional single processor Lanczos run. Unlike other domain decomposition methods, RDMODES does not require DMP.

Analysis	**Comp. Method**	**Submittal Command**	**Suggested Model Type**
Normal modes analysis: **SOL 103**	**RDMODES**	parallel=*p*, nrec=*n*	Large models, allow approximate solution
Modal frequency response analysis: **SOL 111**	**RDMODES +Serial frequency calculation**	parallel =*p*, nrec=*n*	Large models

The DMP computational methods are:

Analysis	**Comp.Method**	**Submittal Command**	**Suggested Problem Type**
Linear Static Analysis: **SOL 101**	**GDSTAT**	dmp=*p*	Large models
	LDSTAT	dmp=*p*, dstat=1	Small models with large number of loads
Normal modes analysis: **SOL103**	**GDMODES**	dmp =*p*	Large models, small frequency range request, also support buckling (SOL 105)
	FDMODES	dmp =*p*, numseg=*p*	Small models, large frequency range request
	HDMODES	dmp =*p*, nclust=*c*	Large models, large frequency range
	RDMODES	dmp =*p*, nrec=*n*	Large models, allow approximate solution
Direct frequency response analysis: **SOL 108**	**FDFREQR**	dmp =*p*	All models

The DMP computational methods with application of normal modes computation are:

Analysis	**Modal Comp. Method**	**Additional Comp. Method**	**Submittal Command**
Modal frequency analysis: **SOL111** Modal transient analysis: **SOL112** Design optimization: **SOL 200**	**GDMODES**	If **SOL 111: FDFREQR** If **SOL 112**: **Serial** transient calculation If **SOL 200**: **Serial** optimization process	dmp =*p*
	FDMODES		dmp =*p*, numseg=*p*
	HDMODES		dmp =*p*, nclust=*c*
	RDMODES		dmp =*p*, nrec=*n*

Note that RDMODES deactivates the sparse eigenvector recovery option in SOL 200. The suggested problem type for a particular method is the same as in SOL 103.

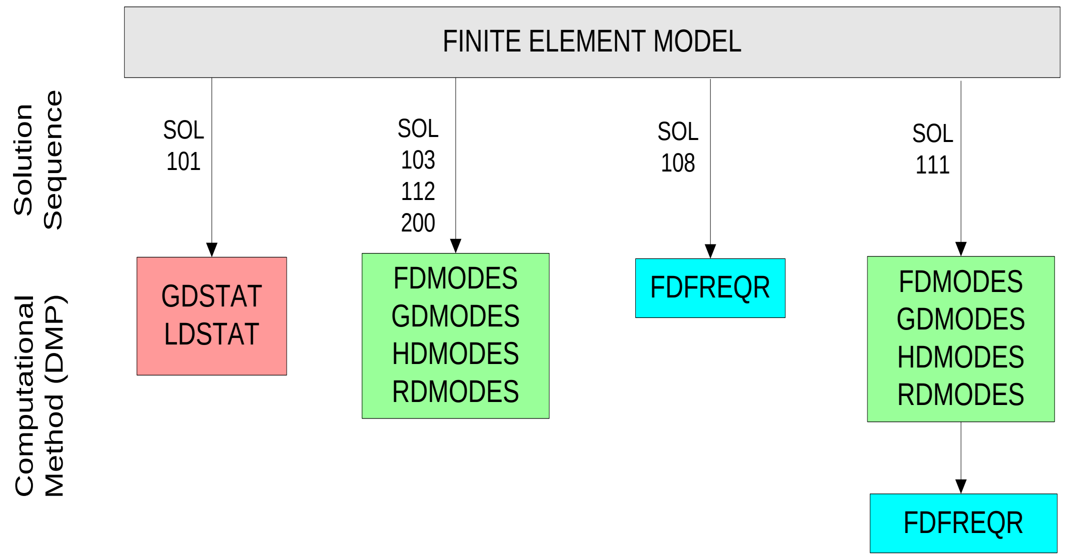
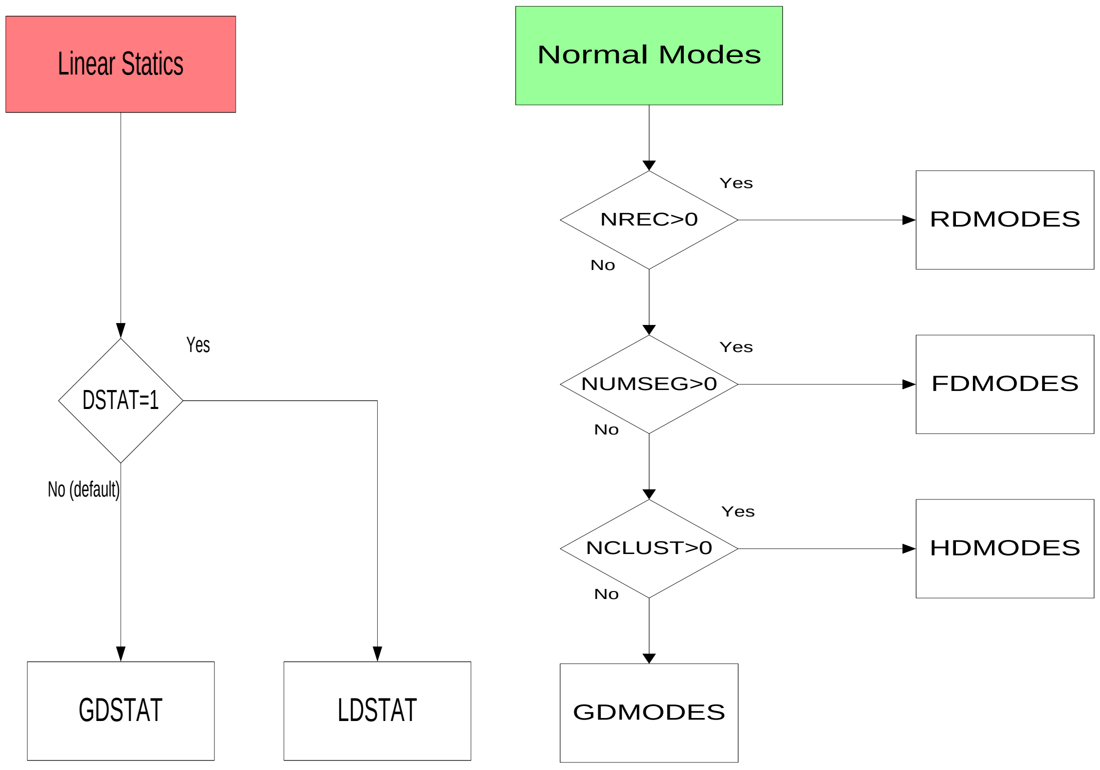


Figure 2.1 Overview of NX Nastran DMP solution tasks

In Figure 2.1, the method selection for SOL 101 is described in Section 3.3. The method selection for DMODES is described in Section 4.5. Overview is summarized as follows.



2.2 Running SMP Jobs

The following example illustrates how to run an NX Nastran job named ‘example.dat’ in an SMP environment with *p* processors.

```
nastran example parallel=p
```

In SOL 103 and SOL 111, one also can specify RDMODES for large problems as follows:

```
nastran example parallel=p nrec=n
```

Where *n* is the number of external components. Refer to RDMODES (section 4.4) for how to choose the number of components.

2.3 Environment Setup for DMP

System Prerequisites

NX Nastran supports the following systems:

X86_64 Linux and Windows-64

The detailed requirements of hardware and software are described in Chapter 7 - Installation and Configuration of DMP, and in the *NX Nastran Installation and Operation Guide*.

Message Passing Prerequisites

NX Nastran uses a Message Passing Interface (MPI) to manage a DMP task. Each compute node must be able to access its local data. It is also necessary to communicate between compute nodes. For these purposes, each local node (host) must have:

- **NX Nastran installed properly**. NX Nastran must be properly installed on all the hosts listed by the "hosts" keyword or in the ‘host.list’ file.
- **MPI program available**. The MPI program start command must be available in the path of the local host. (for example, "mpirun”)
- **Input data file** (including all bulk data and include files) **must be accessible on the local host**.
- **Do NOT assign output file names in data file**.
- **"*r-*" commands available and configured properly**. such as *rsh*, *rcp*, and *rlogin* to communicate between nodes.
- **“scp” and “ssh” are supported on Linux**. Need to put “s.rcp=scp” and “s.rsh=ssh” and set the environment variable MPI_REMSH=ssh. These “s.” commands can be in the command line or in the nastran.ini file.

More details are described in Chapter 7 - Installation and Configuration of Distributed Memory Parallel (DMP).

2.4 Running DMP Jobs

2.4.1 Quick Start

When running an NX Nastran DMP job, it is necessary to specify a computational method for desired solution sequence and a host list of the compute nodes. The Nastran keywords **DMP** and **HOSTS** are required in a DMP run. See section 2.1 for keywords to activate a particular DMP computational method.

```
nastran example [computational method] [host list]
```

Here is an example to run GDMODES with 4 processors for a SOL 103 example, called ‘example.dat’, on a four-node DMP machine. Assume these nodes are named node1, node2, node3 and node4.

```
nastran example dmp=4 hosts=node1:node2:node3:node4
```

Notes:

- The "master" node is the first computer named by the "hosts" keywords, and "slave" nodes are the remaining systems.
- The nastran keywords are processed in both the local and master/slave system.
- It is strongly recommended to use the nastran keyword "slaveout=yes" to print out analysis procedures from all processors. With "slaveout=yes", the f04, .f06, and .log files contain the outputs of master and slave processors.

2.4.2 I/O Enhancement Consideration

The performance of an NX Nastran parallel job is very much dependent on the CPU, memory system, and I/O system performance. A DMP job is extremely sensitive to I/O system performance, since each task independently accesses the I/O system. Especially, the performance of the disk subsystem that contains the permanent and SCRATCH DBSets can have a significant impact on NX Nastran performance. The impact is even greater if multiple tasks are using the same file system.

The scratch directory can be on a global or local file system**.** Siemens PLM recommends that the "sdirectory" be local to each node, if possible.
The following example illustrates how to run ‘example.dat’ in parallel with four processors with "hosts", and "sdirectory" local to each node.

```
nastran example dmp=4 hosts=node1:node2:node3:node4  \
scratch=yes sdirectory=/scr1:/scr2:/scr3:/scr4
```

Note that MIO may be used to reduce the overall run time by using some available computer memory for I/O buffering. The nastran keyword 'mio_cachesize' is used to control such memory size. See the *NX Nastran Installation and Operation Guide.*

2.4.3 Running DMP Jobs with Restart Option

There are two steps to run jobs using the restart option: cold start and restart. The first step, called the cold start, generates the database that will be used in the second step. The restart step runs related jobs with the given database. Note that the first step can be run either in serial or DMP, while the restart step can be run only in serial.

Here is an example where the cold start is a SOL 103 job running DMP on 2 processors, and restart is a SOL 103 example 'exampler.dat' on node1. Assume that the 2 processors are named node1 and node2.

```
Cold Start:
 nastran example dmp=2 hosts=node1:node2 dbs=example

Restart:
 nastran exampler dbs=example.t0
```

Note that the database name in cold start is *example*. Since the cold start is a DMP run, the database saved on the master node (to be used for the restart) will be labeled as *example.t0*. The database *example.t1* from the slave node is not required for restart. Hence for restart jobs, the given database is *example.t0*.

2.4.4 Other Considerations

- Memory (memory) – additional memory usually benefits DCMP; the memory usage high water generally occurs in the partitioning module (GPARTN or GPARTNS) for large models. Over-allocation should be avoided in order to have better performance. Generally, allocated memory should be less than 80% of available physical memory (as a rule of thumb, 50% or less for best I/O performance on Linux systems).
- Scratch (sscr) – Adequate scratch space should be provided.
- SMP and MIO could affect the performance of a DMP job.
- Restarts are supported for a SOL 103 cold start in DMP and serial 103 and 111 restart.
- Contact conditions can be included in SMP and RDMODES runs. See Recursive Domain Normal Modes Analysis (RDMODES).
- Contact conditions cannot be included in GDMODES, FDMODES, or HDMODES runs.

Chapter 3 - Methods for Linear Static Analysis

3.1 Geometric Domain Static Analysis (GDSTAT)

```
nastran example dmp=p
```

The GDSTAT provides an efficient parallel solution for the linear static analysis of large models. The static solution is performed on the *l*-set. The *l*-set is identical to the *a*-set if there is no rigid body support (*r*-set). The mathematical expression for the static equilibrium of the finite element model can be expressed as:

$$K_{ll}u_l = P_l$$

where K_{ll} is the global stiffness matrix, P_l is the load, and u_l is the displacement.

The finite element model in the GDSTAT method is automatically partitioned into *p* domains, where *p* is the number of processors. Figure 3.1 shows each domain that contains the portion of the geometry (O_1, O_2, O_3, *or* O_4) plus the boundary *t*. Think of geometric decomposition as an automated super-element approach. After the domain is partitioned, each processor performs the linear static analysis only on its local domain with boundary. GDSTAT allows a problem that is not possible to solve on one processor to be solved in the DMP environment by reducing a significant amount of disk space and memory.

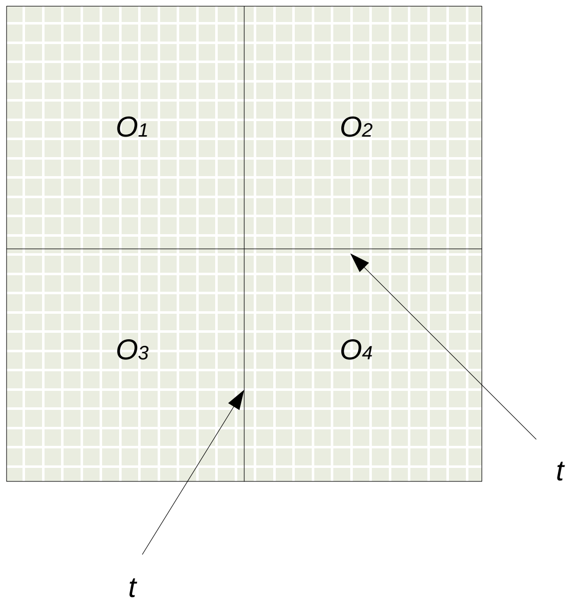


Figure 3.1 Partitioning a finite element model for geometry domain static analysis (GDSTAT)

In terms of partitioning, GDSTAT is available with both the GPARTN module and the SEQP module. The GPARTN module is chosen by default with gpart=1 (the default value). The module SEQP is activated by setting gpart=0. Note that the GPARTN-based GDSTAT does not handle rigid body support (r-set). To avoid rigid body support, constraints should be added using SPC rather than SUPORT cards.

The following example applies to the GPARTN module only. An example for the SEQP module will be given in the appendix.

Example:

```
nastran example dmp=4
```

The static analysis is performed for a finite element model that has 2635 grid points for 200 CHEXA elements and 1210 CTETRA elements. GDSTAT is implemented in a DMP run. The module GPARTN partitioned the global finite element model into four sub-domains and a boundary. The following information in the .f06 describes the detailed statistics of partitioning for the 1st subdomain.

```
     GLOBAL NUMBER OF SHARED ROWS            :        681

     LOCAL NUMBER OF SHARED ROWS             :        342

     LOCAL NUMBER OF SHARED EXCLUSIVE ROWS   :        182

     DESIRED SHARED EXCL. ROWS PER PROCESSOR:         171
```

Each processor performs static analysis with the corresponding local domain and the boundary, and the master processor collects the results that you requested through communications. These outputs are printed only on the master processor.

```
                                         D I S P L A C E M E N T   V E C T O R

      POINT ID.   TYPE          T1             T2             T3             R1             R2             R3
           101      G      3.577434E-10   8.877456E-03  -6.229324E-09   0.0            0.0            0.0
           102      G     -7.238482E-03   9.248573E-03  -2.078553E-03   0.0            0.0            0.0
           103      G      7.267472E-03   1.094641E-02   1.412802E-03   0.0            0.0            0.0
           104      G      1.680989E-09   1.056371E-02  -6.529265E-09   0.0            0.0            0.0
           105      G     -7.267469E-03   1.094641E-02  -1.412815E-03   0.0            0.0            0.0
           106      G      7.455581E-03   1.201167E-02   8.828554E-04   0.0            0.0            0.0
           107      G      6.009865E-10   1.160408E-02  -6.159031E-09   0.0            0.0            0.0
           108      G     -7.455579E-03   1.201167E-02  -8.828678E-04   0.0            0.0            0.0
           109       G       7.623844E-03    1.262617E-02    4.984383E-04    0.0            0.0            0.0
```

It is strongly recommended to use the nastran keyword "slaveout=yes" to print out analysis procedures from all processors. With "slaveout=yes", the f04, .f06, and .log files contain the outputs of master and slave processors in the following format.

```
 ⋮
 } master processor
                                  * * * END OF JOB * * *

****************
  S L A V E   1
 ****************
                                                  ⋮
 }  slave 1 processor
                                  * * * END OF JOB * * *

 ****************
  S L A V E   2
 ****************
                                                  ⋮
 }  slave 2 processor
                                  * * * END OF JOB * * *

 ****************
  S L A V E   3
 ****************
                                                  ⋮
 }  slave 3 processor
                                  * * * END OF JOB * * *
```

3.2 Load Domain Static Analysis (LDSTAT)

```
nastran example dmp=p DSTAT=1
```

LDSTAT is useful when there are a large number of load cases in the linear static analysis problem $K_{ll}u_l = P_l$. Instead of partitioning the finite element model, the load matrix P_l is partitioned among the processors as evenly as possible, and the linear solution is calculated within each of the respective processors for its own load cases. The mathematical expression for the linear static analysis can be expressed as:

$$K_{ll}u_k = P_k, \qquad k = 1, \cdots, p$$

where p is the number of processors. Once all processors finish their own linear solutions, the master processor collects and forms the overall solution as

$$u_l = [u_1, u_2, \cdots, u_p]$$

Note that each processor contains the full model, so that finite element model partitioning is not required. LDSTAT is applied to a large number of loads with the same boundary condition, i.e. only one stiffness matrix K_{ll} is solved in LDSTAT.

Example:

`nastran example dmp=4 dstat=1`

For a model with 500 subcases, 125 subcases are assigned to each processor when four processors are available. Without partitioning into domains, each processor performs the static analysis with the partitioned load cases. The outputs are printed only on the master processor after the master processor collects the requested output results from the slaves

```
                                                                                                                   SUBCASE
101

                                                    D I S P L A C E M E N T   V E C T O R

 POINT ID.   TYPE          T1             T2             T3             R1             R2             R3
  151000      G      3.392543E-07   1.089757E-21   1.783046E-08    .0           -1.204824E-07  -3.644054E-2
  151010      G      3.392543E-07   1.426177E-21  -1.783046E-08    .0            1.204824E-07  -1.336866E-2


                                                                                                                   SUBCASE
102

                                                    D I S P L A C E M E N T   V E C T O R
 POINT ID.   TYPE          T1             T2             T3             R1             R2             R3
  150205      G     -6.509690E-07  -2.062356E-07   8.779039E-22    .0           -1.718401E-21  -1.224331E-
07
  150206      G     -5.072370E-07  -1.837194E-07  -4.060963E-08    .0            1.252286E-07   2.982923E-08



                                            ⋮



                                                                                                                   SUBCASE
500
```

```
                                 D I S P L A C E M E N T   V E C T O R
 POINT ID.   TYPE          T1             T2             T3             R1             R2             R3
  150205      G     -1.303690E-06  -6.062316E-08   9.779011E-20    .0            -3.418412E-22  -8.223322E-
06
  150206      G     -2.172310E-06  -1.937191E-06  -3.060233E-05    .0             4.652283E-08   6.975915E-06
```

3.3 Recommendations for the Method Selection

SOL 101 has two DMP methods for static analysis. It is recommended to use GDSTAT if you have a very large model. LDSTAT is useful when there are large numbers of load cases and a relatively small model.

The selection between GDSTAT and LDSTAT depends on the DSTAT keyword. The method selection is described in Figure 3.2. The default value of DSTAT is 0.

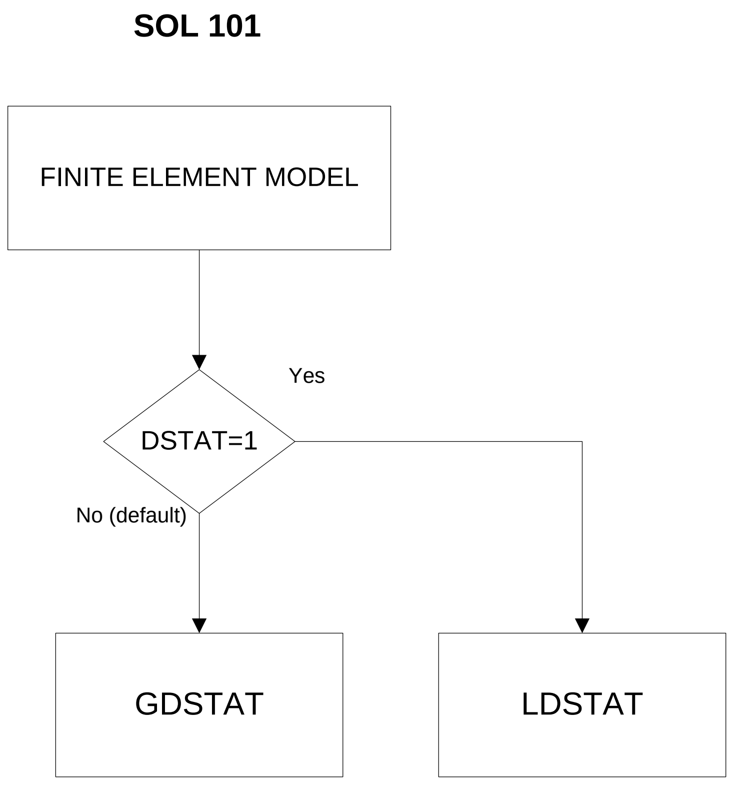


Figure 3.2 DMP Linear Static Analysis (SOL 101)

Chapter 4 - Methods for Normal Modes Analysis

4.1 Geometric Domain Normal Modes Analysis (GDMODES)

```
nastran example dmp=p
```

GDMODES is executed by automatically subdividing the geometry obtained from the finite element model. Such a subdivision of a finite element model is shown in Figure 4.1. Here the *o* partition refers to the interior of the domains and the *t* partition is the common boundary shared by the domains.

The geometric domain decomposition is mathematically represented with the following reordering for the *p* partitioned domains.

$$\begin{bmatrix} K_{oo}^{1}-\lambda M_{oo}^{1} & & & & K_{ot}^{1}-\lambda M_{ot}^{1} \\ & K_{oo}^{2}-\lambda M_{oo}^{2} & & & K_{ot}^{2}-\lambda M_{ot}^{2} \\ & & \ddots & & \vdots \\ & & & K_{oo}^{p}-\lambda M_{oo}^{p} & K_{ot}^{p}-\lambda M_{ot}^{p} \\ K_{oo}^{T,1}-\lambda M_{oo}^{T,1} & K_{oo}^{T,2}-\lambda M_{oo}^{T,2} & \cdots & K_{oo}^{T,p}-\lambda M_{oo}^{T,p} & K_{tt}-\lambda M_{tt} \end{bmatrix} \begin{bmatrix} \phi_o^1 \\ \phi_o^2 \\ \vdots \\ \phi_o^p \\ \phi_t \end{bmatrix} = 0$$

This partitioned eigenvalue problem may be solved by a special formulation of the Lanczos method. Once each processor contains the portion of the local domain with boundary, each processor computes its part of the global eigensolution and exchanges boundary components with other processors.

Table 4-1 shows the eigensolution obtained from each processor, in which the global eigensolutions are obtained by merging all distributed eigensolutions.

Notes:

- The formulation yields a computationally exact solution. The efficiency depends on the relative size of the boundaries with respect to the interiors.
- The geometric domain decomposition is important to reduce the very large problem sizes, but it does not affect the frequency spectrum.
- It is important to use the EIGRL card, not EIGR. In addition, the number of processors *p* should be greater than 1. The value of the keyword "dmp" must be an integer greater than or equal to 2, and a power of 2. For example, 2, 4, 8, 16, etc. are valid.

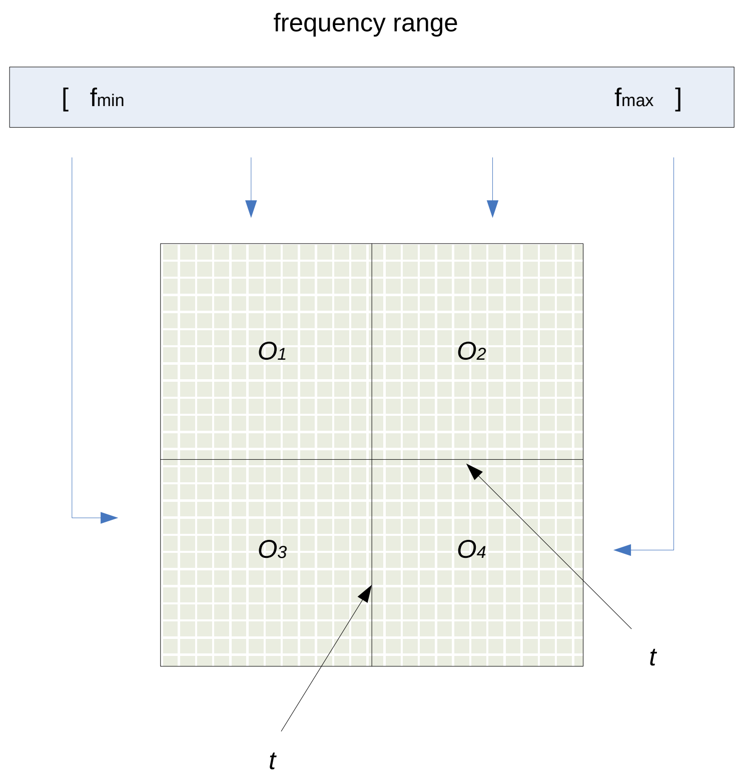


Figure 4.1. Geometry domain partitioning for the normal mode analysis of a finite element model

Table 4-1 The distribution of eigensolution (Φ, Λ) with GDMODES parallel run, where $\Phi = \{\Phi^1, \Phi^2, \cdots, \Phi^p\}^T$.

Geometry domain	partition 1	(Φ^1, Λ)
	$\vdots$	$\vdots$
	partition i	(Φ^i, Λ)
	$\vdots$	$\vdots$
	partition p	(Φ^p, Λ)

In terms of partitioning, GDMODES uses GPARTN by default (see appendix). The module GPARTN performs both degree of freedom-based and grid-based partitioning, in which *p* domains are created by an automatic partitioner from the connectivity graph of the model. PARAM OLDSEQ can be used to specify the desired ordering method.

OLDSEQ	Description
10	Metis with super-nodal and grid compressions
11	MLV with super-nodal and grid compressions (default)
110	Metis with super-nodal compression
111	MLV with super-nodal compression
210	Metis with grid compression
211	MLV with grid compression

Notes:

1. If both super-nodal and grid compressions are selected (OLDSEQ=10 or 11), then the GPARTN returns the coloring with smallest boundary.
2. The default OLDSEQ value is 11. System (294) = 1 prints additional diagnostic information to the f06 file.
3. Note that in some cases, grid compression produces much smaller boundary size than supernodal compression. As a result, the eigensolver (READ module) in GDMODES can have a large run time difference.

Example:

`nastran example dmp=4`

A finite element plate model with 110 grid points and 100 CQUAD4 elements is executed in parallel with SOL 103 analysis. The total number of degrees of freedom is 660. Four processors are used for the DMP run.

The partitioning statistics from the GPARTN module are shown as:

```
 RESULT OF SESET PARTITIONS:

 TOTAL PARTITIONS  TOTAL GRIDS TOTAL BDY. GRIDS  MOVES
             4          496              138       0
 PARTITION    INTERIOR GRIDS    BDY. GRIDS  FRACTION
       1             88              74     .8409
       2             91              74     .8132
       3             88              74     .8409
       4             91              74     .8132

GEOMETRY DOMAIN PARALLEL LANCZOS METHOD
```

Note that the output is related to the degrees of freedom (DOF), not grid points, even though the output uses GRIDS terminology instead of DOF. This example has 88 internal degrees of freedom and 74 boundary degrees of freedom for domain 1. There are 91 and

74, 88 and 74, 91 and 74 degrees of freedom for the interior and boundary of domain 2, 3, and 4, respectively.

The .f06 file of the master processor prints the summary of eigenvalue analysis.

```
EIGENVALUES FOUND IN DOMAIN # 1


     E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)


              BLOCK SIZE USED ......................    7

              NUMBER OF DECOMPOSITIONS ............    3

              NUMBER OF ROOTS FOUND ...............  208

              NUMBER OF SOLVES REQUIRED ...........   41

$--------------------------------------------------------------------------------
$          The list of eigensolutions collected from the slave processor
$--------------------------------------------------------------------------------


                       R E A L   E I G E N V A L U E S
   MODE    EXTRACTION    EIGENVALUE     RADIANS      CYCLES    GENERALIZED  NERALIZED
    NO.       ORDER                                              MASS       STIFFNESS
     1          1          1.696349E+07     4.118676E+03    6.555076E+02   1.000000E+00    1.696349E+07
     2          2          1.848026E+07     4.298867E+03    6.841859E+02   1.000000E+00    1.848026E+07
 ⋮
     207       207         3.073819E+09     5.544203E+04    8.823873E+03   1.000000E+00    3.073819E+09
     208       208         3.123167E+09     5.588530E+04    8.894422E+03   1.000000E+00    3.123167E+09
```

However, with the gpart=1 option, the master processor does not broadcast the collected eigenvalues and/or eigenvectors to the slave processors, so no eigensolutions are printed in the.f06 file except the eigenvalue summary.

```
 ⋮
****************
  S L A V E   1
 ****************
 ⋮
208  EIGENVALUES FOUND IN DOMAIN # 2


        E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)



                    BLOCK SIZE USED ......................    7

                    NUMBER OF DECOMPOSITIONS ............    3

                    NUMBER OF ROOTS FOUND ...............  208

                    NUMBER OF SOLVES REQUIRED ...........   41
```

```
****************
  S L A V E   2
****************
⋮
208  EIGENVALUES FOUND IN DOMAIN # 3

         E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)


                   BLOCK SIZE USED .....................    7

                   NUMBER OF DECOMPOSITIONS ............    3

                   NUMBER OF ROOTS FOUND ...............  208

                   NUMBER OF SOLVES REQUIRED ...........   41


 ****************
  S L A V E   3
 ****************
⋮
208  EIGENVALUES FOUND IN DOMAIN # 4


        E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)

                  BLOCK SIZE USED .....................    7

                  NUMBER OF DECOMPOSITIONS ............    3

                  NUMBER OF ROOTS FOUND ...............  208

                  NUMBER OF SOLVES REQUIRED ...........   41
```

4.2 Frequency Domain Normal Modes Analysis (FDMODES)

```
nastran example dmp=p numseg=p
```

To analyze the dynamic behavior of structures, modal solution techniques are commonly used. The essence of the modal solution is efficient calculation of the mechanical system's free, undamped vibration. That is the eigenvalue analysis problem of:

$$(K_{aa} - \lambda M_{aa})\phi_a = 0$$

Here the eigenvalue λ represents a natural frequency, and the eigenvector ϕ is a free vibration shape of the finite element model. The a subscript refers to the *a*-set, which is the analysis partition of the finite element model. This is one of the most time consuming computations of large scale global analyses in the automobile and the aerospace industries.

The frequency range of interest specified on the EIGRL entry is automatically decomposed into multiple frequency segments—one for each processor. Table 4-2 shows the frequency segment of each processor when *s* processors are available.

Table 4-2 Partition of frequency for FDMODES parallel run

Processor	**Frequency segment**	**Lower frequency**	**Upper frequency**
1	1	f_0	f_1
2	2	f_1	f_2
$\vdots$	$\vdots$		
j	j	f_{j-1}	f_j
$\vdots$	$\vdots$		
s	s	f_{s-1}	f_s

Note that each processor contains the full model in the FDMODES computation, so that the mode shapes in the individual frequency segments are independent of each other. Figure 4.2 represents the schematic diagram: each processor solves the full model within its frequency segment. The only communication needed is when gathering the results for the master processor. Table 4-3 shows the eigensolutions obtained from each processor.

Notes:

- Although FDMODES reduces the frequency range for a process by decomposing the frequency domain, it is still ineffective with respect to large problem size.
- Each processor contains the full model, so that finite element model partitioning is not required. For best load balance, V1 and V2 of EIGRL should be specified and ND omitted.

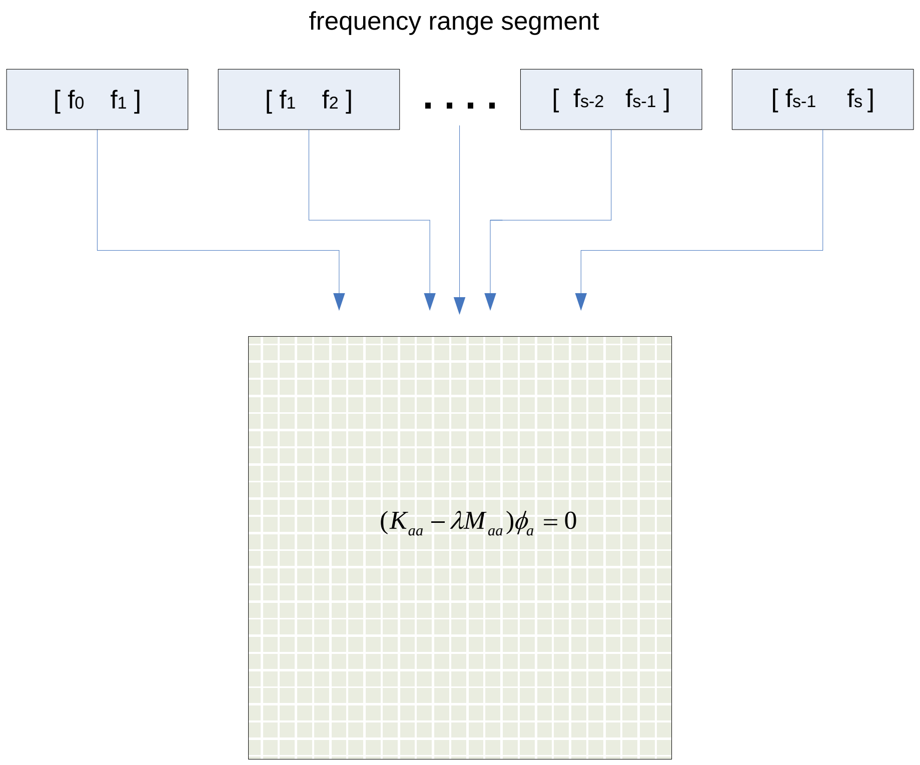


Figure 4.2 Frequency domain partitioning for the normal mode analysis of a whole finite element model

Table 4-3. The distribution of eigensolution $(\Phi, \Lambda), \Phi = \{\Phi_1, \Phi_2, \cdots, \Phi_s\}$ and $\Lambda = diag\{\Lambda_1, \Lambda_2, \cdots, \Lambda_s\}$, with FDMODES parallel run

Frequency domain				
Segment 1	…	Segment *j*	…	Segment *s*
(Φ_1, Λ_1)	…	(Φ_j, Λ_j)	…	(Φ_s, Λ_s)

Example:

```
nastran example dmp=4 numseq=4
```

This is the example used in GDMODES. FDMODES does not require domain decomposition. The .f06 file shows the number of eigenvalues calculated in each processor. Each processor computes 94, 61, 26, and 27 eigenvalues, respectively. The total number of modes found is 208.

The master processor collects the eigenvalues and eigenvectors of slave processors, and prints out the merged eigensolutions.

```
      FREQUENCY DOMAIN PARALLEL LANCZOS METHOD

 94  EIGENVALUES FOUND IN DISTRIBUTED SEGMENT # 1

 61  EIGENVALUES FOUND IN DISTRIBUTED SEGMENT # 2

 26  EIGENVALUES FOUND IN DISTRIBUTED SEGMENT # 3

 27  EIGENVALUES FOUND IN DISTRIBUTED SEGMENT # 4

   E I G E N V A L U E   A N A L Y S I S   S U M M A R Y   (READ MODULE)

              BLOCK SIZE USED ......................    7

              NUMBER OF DECOMPOSITIONS ............    5

              NUMBER OF ROOTS FOUND ...............  208

              NUMBER OF SOLVES REQUIRED ...........   45

$--------------------------------------------------------------------------------
$          The list of all eigenvalues collected from the slave processor
$--------------------------------------------------------------------------------

                                 R E A L   E I G E N V A L U E S
MODE    EXTRACTION       EIGENVALUE       RADIANS      CYCLES      GENERALIZED     GENERALIZED
 NO.       ORDER                                                       MASS          STIFFNESS
 1           1       1.696349E+07      4.118676E+03      6.555076E+02      1.000000E+00     1.696349E+07
 2           2       1.848026E+07      4.298867E+03      6.841859E+02      1.000000E+00     1.848026E+07
 ⋮
207         207      3.073819E+09      5.544203E+04      8.823873E+03      1.000000E+00     3.073819E+09
208         208      3.123167E+09      5.588530E+04      8.894422E+03      1.000000E+00     3.123167E+09
```

With "slaveout=yes", you can see the information about the eigenvalue problem on each slave processor. It also shows the number of eigenvalues found on the slave processors.

```
****************
  S L A V E   1
 ****************

  FREQUENCY DOMAIN PARALLEL LANCZOS METHOD
                                  ⋮
EIGENVALUES FOUND IN DISTRIBUTED SEGMENT # 2


****************
  S L A V E   2
 ****************
```

```
  FREQUENCY DOMAIN PARALLEL LANCZOS METHOD
                                 ⋮
EIGENVALUES FOUND IN DISTRIBUTED SEGMENT # 3



****************
  S L A V E   3
 ****************

  FREQUENCY DOMAIN PARALLEL LANCZOS METHOD
                                  ⋮
EIGENVALUES FOUND IN DISTRIBUTED SEGMENT # 4
```

4.3 Hierarchic Domain Normal Modes Analysis (HDMODES)

```
nastran example dmp=p nclust=c
```

The HDMODES scheme simultaneously combines the two previously presented methods, GDMODES and FDMODES. With this approach, a subset of processors or a cluster solves the eigenvalue problem for the local geometry while communicating with other subsets of processors or other clusters in order to consider the other frequency ranges.

The HDMODES computation is based on the processor assignment shown in Table 4-4, where we have *p* * *s* processors available. For each frequency segment, *p* processors are used for GDMODES. In other words, *s* processors are used for FDMODES for each geometric portion. For example, the ((*j* - 1) *p*+*i*)-th processor computes the eigenvalues of the j-th frequency segment Λ_j and the i-th geometric partition of the corresponding eigenvectors Φ^i_j. The selection of the *s* and *p* value is problem dependent.

Note that HDMODES solves the eigenvalue problem computationally exactly, just as the GDMODES and FDMODES methods do.

Table 4-4 Hierarchic domain decomposition concept

		Frequency domain				
		segment 1	…	segment *j*	…	Segment *s*
Geometry domain	partition 1	1 (Φ^1_1, Λ_1)		*(j-1)*p+1* (Φ^1_j, Λ_j)		*(s-1)*p+1* (Φ^1_s, Λ_s)
	⋮	⋮		⋮		⋮
	partition *i*	*i* (Φ^i_1, Λ_1)		*(j-1)*p+i* (Φ^i_j, Λ_j)		*(s-1)*p+i* (Φ^j_s, Λ_s)
	⋮	⋮		⋮		⋮
	partition *p*	*p* (Φ^p_1, Λ_1)		*j*p* (Φ^p_j, Λ_j)		*s*p* (Φ^p_s, Λ_s)

The preferred hardware environment for HDMODES is a cluster of multiprocessor workstations that is usually tied together by either a hardware switch or a network, as illustrated in Fig. 4.3.

Assuming *m* workstations with *n* processors each, based on the scheme of Table 4-4, the tasks of each column (the geometric partitions of a particular frequency segment) reside on one workstation. The tasks of each row (the various frequency segments of a particular geometry partition) are spread across the cluster, as shown in Fig. 4.4.

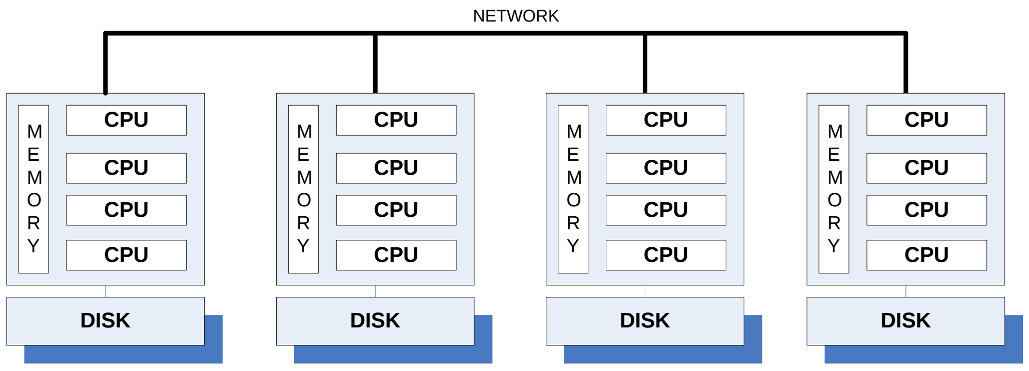


Figure 4.3. A scheme of cluster of multi-processor workstations

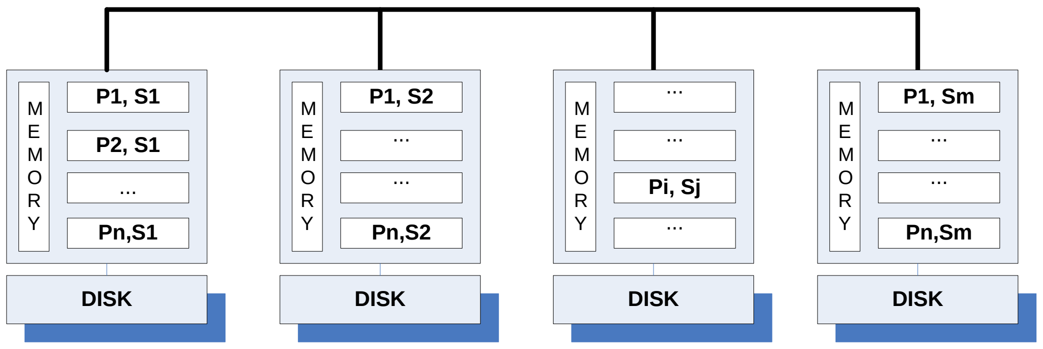


Figure 4.4. Mapping the hierarchic domain decomposition with *n* geometric domain partition and *m* frequency partition (**Pi**: geometric domain partition and **Sj**: frequency segment) to a cluster of multi-processor workstations.

The HDMODES solution sequence combines two existing techniques of DMP processing: GDMODES and FDMODES. While the geometry partitions are solved within a set of processors called a cluster, frequency segments are also solved in parallel across multiple clusters. The advantage is a faster eigenvalue problem solution time for very large models.

The keyword "dmp" defines the number of processors p, and the keyword "nclust" defines the number of clusters c. The number of geometry partitioning g in a cluster does not have to be defined explicitly. Note that $1<c<p$ and $p=c*g$. The number of clusters c should be properly selected so that g is an integer greater than or equal to 2, and a power of 2. For example, 2, 4, 8, 16, etc. are valid.

Example:

`nastran example dmp=4 nclust=2`

In the example below, HDMODES is executed with ‘dmp=4’ and ‘nclust=2’ keywords. Among the four processors, the first processor is the master processor, and the other processors are the slave processors.

Note that HDMODES defines the local master and local slave processors inside of each cluster.

The partitioning statistics with GPARTN describe the number of degrees of freedom for each cluster. In the example below, the first cluster has 223 degrees of freedom in the interior of domain 1, and 50 degrees of freedom in the boundary. The second cluster has the same local size as the first one.

```
TOTAL PARTITIONS  TOTAL GRIDS TOTAL BDY. GRIDS  MOVES
              2          496              50       0
 PARTITION   INTERIOR GRIDS    BDY. GRIDS  FRACTION
        1              223           50     .2242
        2              223           50     .2242

  HIERARCHIC DOMAIN PARALLEL LANCZOS METHOD
```

In the .f06 file, the master processor prints the summary of the eigenvalue analysis and the list of eigenvalues. Be careful when interpreting the "number of roots found" information in the summary. This information concerns the cluster in which the master is included. For example, in the model below, interpret the "number of roots found" message as indicating that cluster 1 found 155 eigenvalues. The master processor lists all 208 eigenvalues that are merged from all of the local master processors. You should determine the total number of modes from the list of eigenvalues.

```
VALUES FOUND IN DOMAIN # 1

        E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)


                  BLOCK SIZE USED ......................    7

                  NUMBER OF DECOMPOSITIONS ............    3

                  NUMBER OF ROOTS FOUND ...............  155

                  NUMBER OF SOLVES REQUIRED ...........   41

$--------------------------------------------------------------------------------
$           The list of all eigensolutions collected from the slave processor
$--------------------------------------------------------------------------------
```

```
                     R E A L   E I G E N V A L U E S
   MODE    EXTRACTION    EIGENVALUE    RADIANS      CYCLES   GENERALIZED  NERALIZED
    NO.       ORDER                                             MASS       STIFFNESS
    1         1      1.696349E+07  4.118676E+03 6.555076E+02  1.000000E+00       1.696349E+07
    2         2      1.848026E+07  4.298867E+03 6.841859E+02  1.000000E+00       1.848026E+07
 ⋮

    207     207      3.073819E+09  5.544203E+04 8.823873E+03  1.000000E+00       3.073819E+09
    208     208      3.123167E+09  5.588530E+04 8.894422E+03  1.000000E+00       3.123167E+09
```

Again, be cautious when interpreting the output from slave processors. The summary of eigenvalue analysis for slave processors is confined to the corresponding processor. In the list of eigenvalues on the slave processors, the mode number does not represent the global mode number.

With gpart=1, the master process does not broadcast the collected output to slave processors, so no eigenvalues are listed in the information from the slave processors.

```
 ⋮
****************
  S L A V E   1
 ****************
 ⋮
53  EIGENVALUES FOUND IN DOMAIN # 1

        E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)


                  BLOCK SIZE USED ......................    7

                  NUMBER OF DECOMPOSITIONS ............    2

                  NUMBER OF ROOTS FOUND ................    53

                  NUMBER OF SOLVES REQUIRED ...........   40

****************
  S L A V E   2
 ****************
 ⋮
155  EIGENVALUES FOUND IN DOMAIN # 2

          E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)


                    BLOCK SIZE USED ......................    7

                    NUMBER OF DECOMPOSITIONS ............    3

                    NUMBER OF ROOTS FOUND ................  155

                    NUMBER OF SOLVES REQUIRED ...........   41

****************
  S L A V E   3
 ****************
 ⋮
53  EIGENVALUES FOUND IN DOMAIN # 2
```

```
          E I G E N V A L U E   A N A L Y S I S   S U M M A R Y   (READ MODULE)


                   BLOCK SIZE USED ......................     7

                   NUMBER OF DECOMPOSITIONS ............     2

                   NUMBER OF ROOTS FOUND ................    53

                   NUMBER OF SOLVES REQUIRED ...........    40
```

4.4 Recursive Domain Normal Modes Analysis (RDMODES)

```
nastran example dmp=p nrec=m
```

The Recursive Domain Normal Modes (RDMODES) analysis extends the DMP parallel capability via substructuring technology for very large scale normal nodes analysis. It is currently available in SOL 103 and 111. It also supports modal analysis in superelement jobs. The RDMODES approach generally computes fewer modes with lower accuracy compared to standard Lanczos approaches in order to gain performance.

RDMODES begins with partitioning the eigenvalue problem $Kx = \lambda Mx$ into *nrec* external partitions. The following represents a reordering for the nrec=4 with total 7 (=2*nrec-1) components for matrix K.

$$K = \left[\begin{array}{ccc:ccc:c}
K_{oo}^{4} & & K_{ot}^{4,2} & & & & K_{ot}^{4,1} \\
& K_{oo}^{5} & K_{ot}^{5,2} & & & & K_{ot}^{5,1} \\
* & * & K_{oo}^{2} & & & & K_{ot}^{2,1} \\
\hdashline
& & & K_{oo}^{6} & & K_{ot}^{6,3} & K_{ot}^{6,1} \\
& & & & K_{oo}^{7} & K_{ot}^{7,3} & K_{ot}^{7,1} \\
& & & * & * & K_{oo}^{3} & K_{ot}^{3,1} \\
\hdashline
* & * & * & * & * & * & K_{oo}^{1}
\end{array}\right]$$

(here the asterisks denote the transpose of the corresponding blocks in the upper triangular portion, as the matrices are symmetric). Matrix M has the same structure.

This partitioned eigenvalue problem may be solved by a special formulation of the sub-structuring method. All components are distributed evenly into processors. Each interior eigensolution corresponding to its external partition is performed in serial, independent of the others. If the keyword *nclust* is specified, the processors are divided into *n* clusters as in HDMODES. In this case, each interior eigensolution is performed in GDMODES fashion in its own cluster.

Contact conditions can be included in an RDMODES run. The input file should include a static subcase with the contact conditions, and a consecutive normal modes subcase which includes the STATSUB case control command. When you run with RDMODES and contact conditions combined, an automatic static condensation is performed by default during the static portion of the solution such that the contact iterations occur in a reduced representation. As a result, performance gains occur in both the static and the RDMODES portions. To run RDMODES without the static condensation, include the parameter setting PARAM,RDCNT,NO.
The following is not supported when you combine RDMODES and contact conditions:

- Inertia relief, which is defined with the INREL parameter.

- The constraint mode method of enforced motion. The absolute method can be selected instead with the system cell ENFMOTN.

RDMODES Sparse Eigenvector Recovery

In many instances, a user is only interested in the solutions at a few key locations instead of all degrees of freedom, especially for large problems with millions of degrees of freedom. In such cases, the sparse eigenvector recovery method can significantly reduce the overall computation time and storage resource.

In RDMODES, the sparse eigenvector recovery option will be determined automatically based on the user's output request. If full eigenvectors are desired with only few output requests, a user can deactivate sparse data recovery with PARAM,RDSPARSE,NO in the BULK data.

RDMODES with the rdsparse option supports residual vectors (PARAM, RESVEC), panel participation factors (PARAM, PANELMP), absolute displacement enforced motion (sys422=1), and modal contributions. Note that PARAM,RESVINER is not supported.

Note that the accelerated residual vector calculation with RDMODES takes advantage of the rdsparse option, and is more efficient than the original one in terms of computational time and I/O usage. The residual vectors with the accelerated calculation may differ slightly from the original, which cannot be used in conjunction with rdsparse. If necessary, the original resvec method may be requested by specifying PARAM, RDRESVEC, NO in the bulk data. In this case, the rdsparse option will be disabled automatically, which is likely to result in dramatically reduced performance.

Running RDMODES

RDMODES is activated by the Nastran keyword **nrec**. It can run in serial, SMP, and DMP with optional keywords **nclust=*c*** and **rdscale**.

`Serial`	**`Nastran example nrec=`*`n`***
`SMP`	**`Nastran example parallel=`*`p`*` nrec=`*`n`***
`DMP`	**`Nastran example dmp=`*`p`*` nrec=`*`n`***

Notes:

1. *p* is an integer equal to a power of 2. It can be 1 (i.e. *p*=1).

2. The keyword *nrec* must be a positive number. The efficiency of the method requires carefully choosing m, which should not be too small or too large. A nrec value with power of 2 is required for better performance. For a large problem (>1M DOFs) nrec equal to 128 or 256 would be a good choice.

Number of grids	NREC value
1 ~ 5,000	4
5,000 ~ 40,000	8
15,000 ~ 80,000	16
30,000 ~ 150,000	32
60,000 ~ 300,000	64
120,000 ~ 600,000	128
250,000 ~1,200,000	256
> 1,200,000	512

This table does not suggest a unique *nrec* value for a given model. The best choice will depend on the model and on the user's machine configuration.

3. If the keyword *nclust* is used, c should be properly selected so that p/c is an integer greater than or equal to 2 and a power of 2.

4. The optional keyword *rdscale* is used to increase the accuracy of the solution. In most practical circumstances values in the range of 1.0 to 5.0 are acceptable. The trade off is that the computational time increases with higher values of rdscale. The default value is 2.5.

5. PARAM,OLDSEQ in the input file can be used to specify the method of creating the substructures. OLDSEQ is an existing parameter for geometric domain partitioning.

OLDSEQ	Description
10	Metis with super-nodal and grid compressions
11	MLV with super-nodal and grid compressions (default)
110	Metis with super-nodal compression
111	MLV with super-nodal compression
210	Metis with grid compression
211	MLV with grid compression

Example:

```
nastran example dmp=4 nrec=8
```

In the example below, RDMODES is executed with ‘dmp=4’ and ‘nrec=8’ keywords. Among the four processors, the first processor is the master processor, and the other processors are the slave processors.

After the special formulation of the substructuring method, RDMODES calls 4-way FDMODES for the reduced eigenvalue problem to obtain the global eigensolution. The master processor collects the eigenvalues and eigenvectors of slave processors, and prints out the merged eigensolution.

```
FREQUENCY DOMAIN PARALLEL LANCZOS METHOD
⋮
    94  EIGENVALUES FOUND IN DISTRIBUTED SEGMENT #         1

    61  EIGENVALUES FOUND IN DISTRIBUTED SEGMENT #         2

    26  EIGENVALUES FOUND IN DISTRIBUTED SEGMENT #         3

    27  EIGENVALUES FOUND IN DISTRIBUTED SEGMENT #         4

            E I G E N V A L U E  A N A L Y S I S  S U M M A R Y  (READ MODULE)

                 BLOCK SIZE USED ......................   7

                 NUMBER OF DECOMPOSITIONS ............   4

                 NUMBER OF ROOTS FOUND ................  208

                 NUMBER OF SOLVES REQUIRED ...........   43

                     R E A L   E I G E N V A L U E S
                   (BEFORE AUGMENTATION OF RESIDUAL VECTORS)
 MODE   EXTRACTION    EIGENVALUE        RADIANS         CYCLES        GENERALIZED       GENERALIZED
  NO.     ORDER                                          MASS          STIFFNESS
   1       1      1.696475E+07     4.118829E+03     6.555319E+02     1.000000E+00     1.696475E+07
   2       2      1.848272E+07     4.299153E+03     6.842314E+02     1.000000E+00     1.848272E+07

   ⋮

 207     207      3.087263E+09     5.556315E+04     8.843149E+03     1.000000E+00     3.087263E+09
 208     208      3.125765E+09     5.590854E+04     8.898121E+03     1.000000E+00     3.125765E+09
```

With "slaveout=yes", you can see the information about the eigenvalue problem on each slave processor. It also shows the number of eigenvalues found on the slave processors.

```
****************
  S L A V E   1
****************
   FREQUENCY DOMAIN PARALLEL LANCZOS METHOD

   ⋮
         61  EIGENVALUES FOUND IN DISTRIBUTED SEGMENT #           2

****************
  S L A V E   2
****************
   FREQUENCY DOMAIN PARALLEL LANCZOS METHOD

   ⋮
         26  EIGENVALUES FOUND IN DISTRIBUTED SEGMENT #           3

****************
  S L A V E   3
****************
   FREQUENCY DOMAIN PARALLEL LANCZOS METHOD

   ⋮
         27  EIGENVALUES FOUND IN DISTRIBUTED SEGMENT #           4
```

4.5 Recommendations for Method Selection

It is recommended that you use FDMODES if you have a small model and a large frequency range of interest. If you have a very large model and insufficient disk space, use either GDMODES or HDMODES. Especially, HDMODES is recommended for a cluster of multiprocessor workstations. RDMODES is recommended for large frequency range, large models, and best performance when reduced accuracy is acceptable.

Fig. 4.5 illustrates the general guide for selecting a suitable DMP computational method for normal modes analysis.

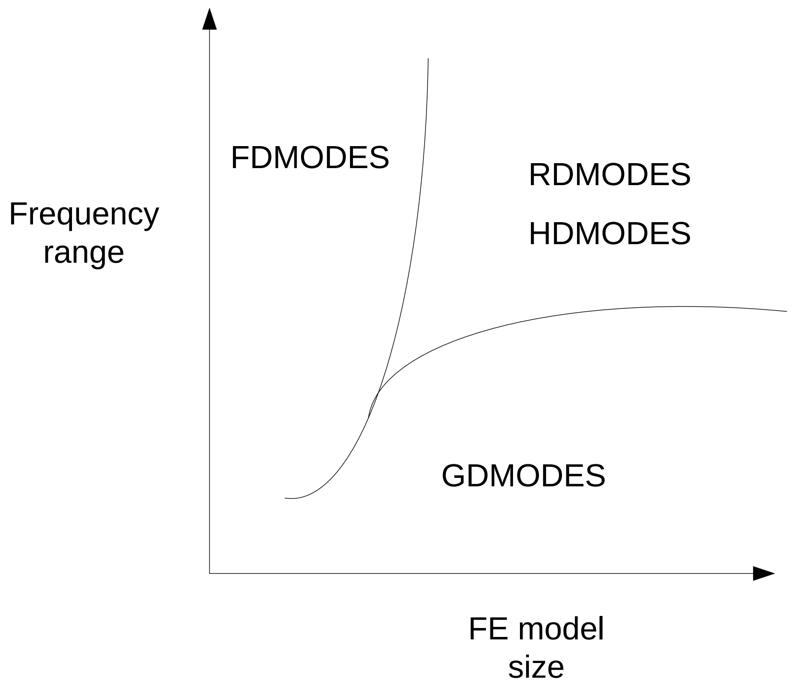


Figure 4.5 General guideline for the selection of DMP computational methods

The selection of DMP computational methods introduced in this chapter is made according to the keywords, NREC, NUMSEG, and NCLUST. The method selection is described in Figure 4.6. Note the default value of NREC, NUMSEG and NCLUST is 0.

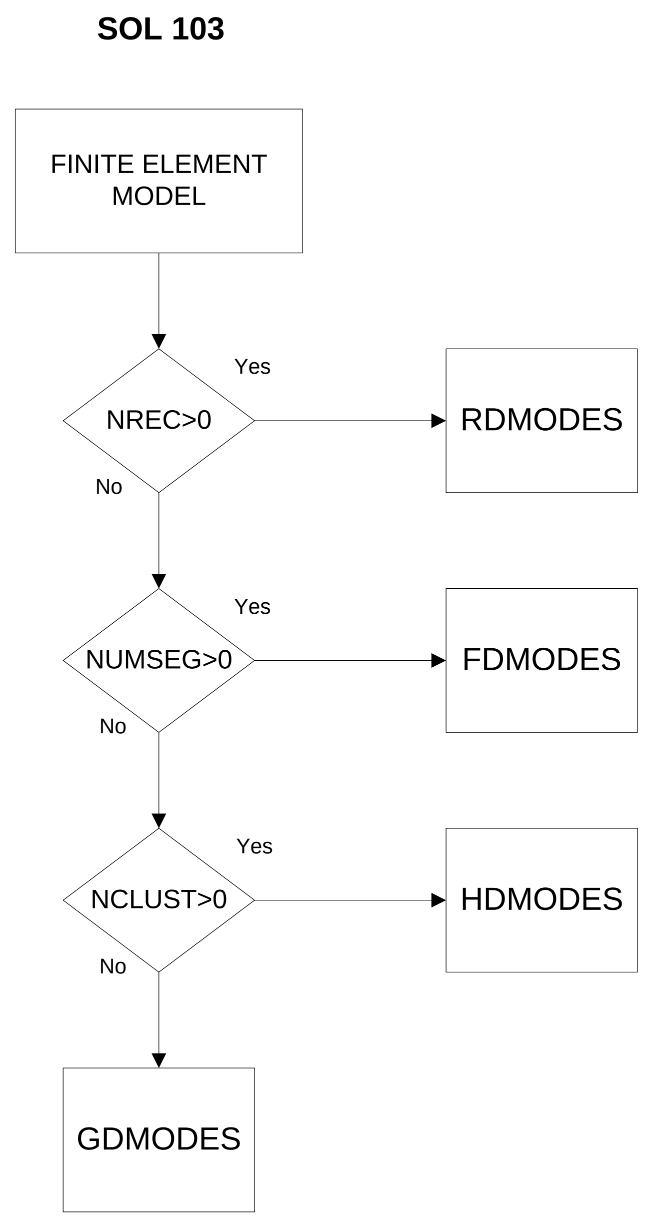


Figure 4.6 DMP normal mode analysis (SOL 103)

Chapter 5 - Methods for Response Analysis and Optimization

5.1 Frequency Domain Frequency Response Analysis (FDFREQR)

```
nastran example dmp=p
```

The excitation frequencies, as specified on the FREQi entries, are split among the processors as evenly as possible, and the responses for the partitioned excitation frequencies are then calculated within each of the respective processors as shown in. Fig.5.1. The whole range of excitation frequencies is partitioned into many sub-intervals such as [F_i F_{i+1}]. Note that each processor contains the full model.

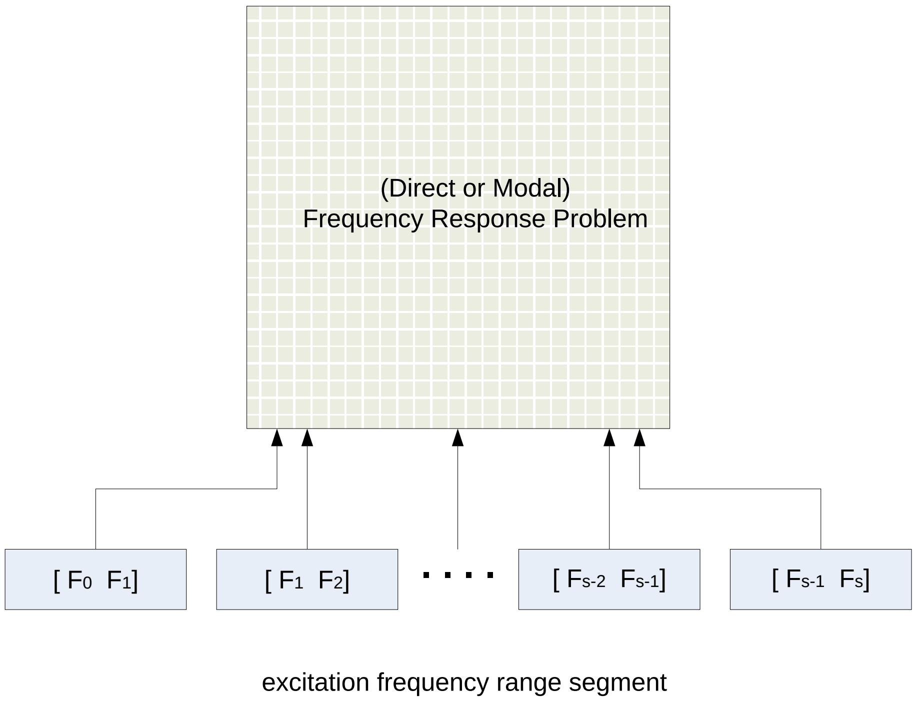


Figure 5.1 Excitation frequency domain partitioning for (direct or modal) frequency response analysis of a whole finite element model

For SOL 108, if there are many forcing frequencies, FDFREQR should be used. In FDFREQR, the resource requirements such as disk space and memory are as large as the resource requirements of a serial run.

Note that, for SOL 111, running a job with GDMODES, FDMODES, or HDMODES automatically performs FDFREQR without requesting it explicitly. Performing FDFREQR partitions the excitation frequencies among the number of processors defined with the "dmp" keyword.

Example: SOL 108

`nastran example dmp=4`

For a finite element plate model that has 110 grid points and 100 CQUAD4 elements, SOL 108 analysis is performed in parallel with the FDFREQR method. The total number of excitation frequencies is 89 in the following entry of FREQ1.

```
FREQ1   10      3.0     3.0     88
```

The .f06 file describes the partitioned frequency range. In the example below, processors 1, 2, 3, and 4 execute 23, 22, 22, and 22 excitation frequency ranges. Once each processor finishes its own analysis, the master processor collects the results and prints the output.

```
 ⋮
     DISTRIBUTED MEMORY PARALLEL FREQUENCY RESPONSE
     NUMBER OF FREQUENCY DOMAINS  =       4
     NUMBER OF FREQUENCIES ON LOCAL PROCESSOR (ID=   1) =       23
 ⋮
$-----------------------------------------------------------------------------
$          The list of all responses collected from the slave processor
$-----------------------------------------------------------------------------

R E S P O N S E   O U T P U T

****************
  S L A V E   1
 ****************
 ⋮
     DISTRIBUTED MEMORY PARALLEL FREQUENCY RESPONSE
     NUMBER OF FREQUENCY DOMAINS  =       4
     NUMBER OF FREQUENCIES ON LOCAL PROCESSOR (ID=   2) =       22
 ⋮
****************
  S L A V E   2
 ****************
 ⋮
     DISTRIBUTED MEMORY PARALLEL FREQUENCY RESPONSE
     NUMBER OF FREQUENCY DOMAINS  =       4
     NUMBER OF FREQUENCIES ON LOCAL PROCESSOR (ID=   3) =       22
 ⋮
****************
```

```
 S L A V E   3
****************
⋮
    DISTRIBUTED MEMORY PARALLEL FREQUENCY RESPONSE
    NUMBER OF FREQUENCY DOMAINS  =       4
    NUMBER OF FREQUENCIES ON LOCAL PROCESSOR (ID=   4) =       22
⋮
```

5.2 DMODES + FDFREQR for SOL 111

In SOL 111, the FDMODES, GDMODES, HDMODES, RDMODES methods will automatically continue to perform FDFREQR after the eigenvalue analysis is finished, so you do not need to request FDFREQR explicitly for the modal frequency response. Therefore, the parallel SOL 111 job can be run in exactly the same ways as the parallel SOL 103. The DMP tasks in SOL 111 are described in Figure 5.2. The method selection of DMP normal mode analysis is similar to that of SOL 103, which was introduced in Figure 4.6.

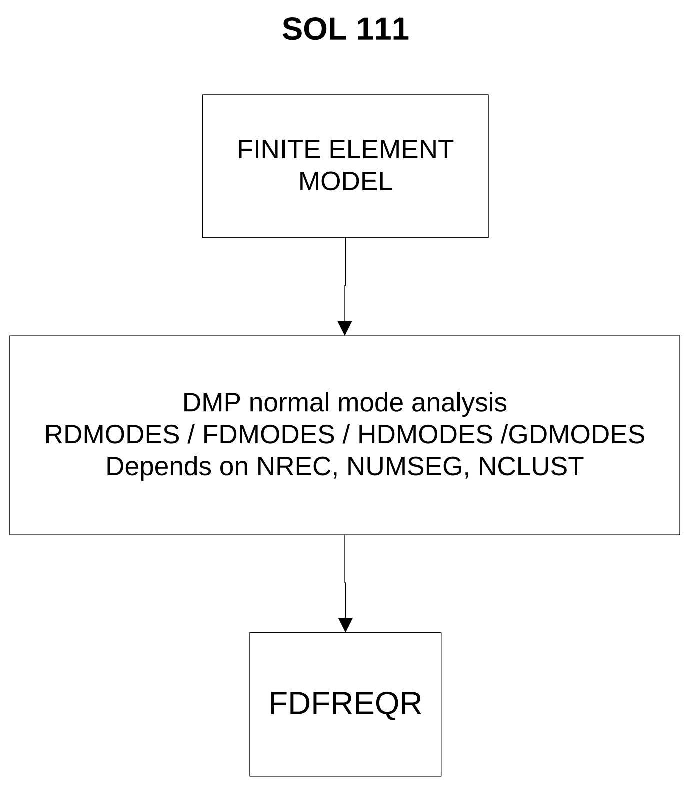

SOL 111
FINITE ELEMENT MODEL
DMP normal mode analysis
RDMODES / FDMODES / HDMODES /GDMODES
Depends on NREC, NUMSEG, NCLUST
FDFREQR

Figure 5.2 DMP modal frequency response analysis (SOL 111)

Example: SOL 111 (FDMODES + FDFRQR)

```
nastran example dmp=4
```

Each processor computes 94, 61, 26, and 27 eigenvalues, respectively, for the whole finite element model. For the modal frequency response analysis, 19 excitation frequencies are split to 5, 5, 5, and 4, and distributed to each processor.

```
 ⋮
$--------------------------------------------------------------------------
$          The number of eigenvalues found at each processor
$--------------------------------------------------------------------------

 94  EIGENVALUES FOUND IN DISTRIBUTED SEGMENT # 1

 61  EIGENVALUES FOUND IN DISTRIBUTED SEGMENT # 2

 26  EIGENVALUES FOUND IN DISTRIBUTED SEGMENT # 3

 27  EIGENVALUES FOUND IN DISTRIBUTED SEGMENT # 4
 ⋮
E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)
 ⋮
$--------------------------------------------------------------------------
$          The number of excitation frequencies partitioned for each processor
$--------------------------------------------------------------------------

     PERFORMANCE SUMMARY TABLE FOR DISTRIBUTED MEMORY FREQUENCY RESPONSE
     NUMBER OF FREQUENCY DOMAINS  =       4
     NUMBER OF FREQUENCIES        =      19
     PROCESSOR                    # FREQ.        CPU (SEC)      ELAPSED (SEC)
       ---------                   -------       ---------      -------------
       1. ugs001                       5            1.57             2.22
       2. ugs002                       5            1.87             2.22
       3. ugs003                       5            1.75             2.22
       4  ugs004                       4            1.49             2.20
 ⋮
$--------------------------------------------------------------------------
$          The list of all responses collected from the slave processor
$--------------------------------------------------------------------------

R E S P O N S E   O U T P U T

 ⋮
****************
  S L A V E   1
 ****************
 ⋮
E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)



****************
  S L A V E   2
 ****************
 ⋮
E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)

****************
  S L A V E   3
 ****************
 ⋮
E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)
```

5.3 DMODES + Serial transient calculation for SOL 112

In the parallel SOL 112 run, once the eigenvalue analysis is performed in parallel, the modal transient response analysis is executed in serial. During the modal transient response analysis, all slave processors are idle. The DMP task SOL112 is described in Figure 5.3. Note that transient response analysis (TRD1 module) does not benefit from DMP. The method selection of DMP normal mode analysis is similar to that of SOL 103, which is introduced in Figure 4.6.

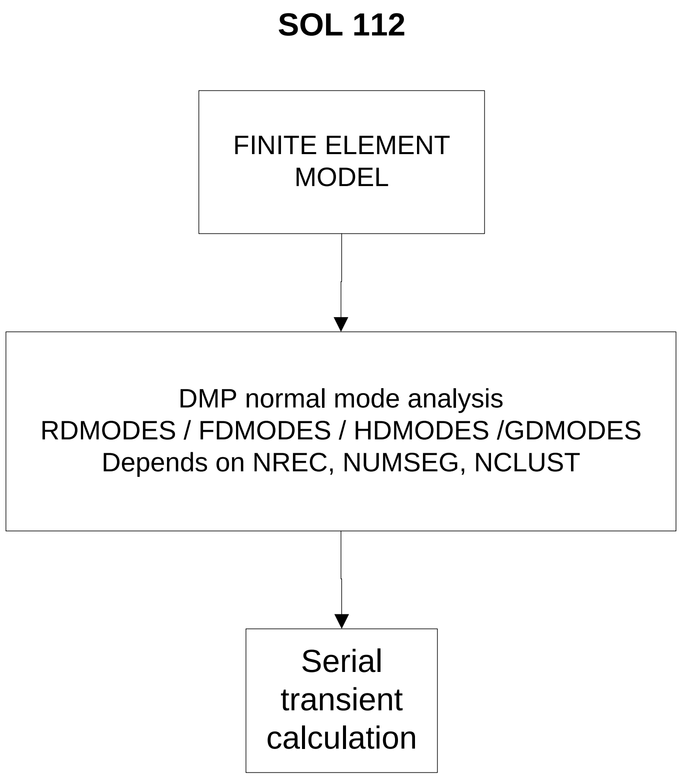

SOL 112
FINITE ELEMENT MODEL
DMP normal mode analysis
RDMODES / FDMODES / HDMODES /GDMODES
Depends on NREC, NUMSEG, NCLUST
Serial transient calculation

Figure 5.3 DMP modal transient response analysis (SOL 112)

Example: SOL 112 (GDMODES + Serial transient calcuation)

```
nastran example dmp=4
```

The output of responses in the .f06 file is printed only on the master processor.

```
 ⋮
E I G E N V A L U E   A N A L Y S I S   S U M M A R Y   (READ MODULE)

 ⋮

$-------------------------------------------------------------------------------
-$          Only the master processor run the transient responses analysis
$-------------------------------------------------------------------------------
-


R E S P O N S E   O U T P U T
 ⋮

****************
  S L A V E   1
 ****************
 ⋮
E I G E N V A L U E   A N A L Y S I S   S U M M A R Y   (READ MODULE)

 ⋮
****************
  S L A V E   2
 ****************
 ⋮
E I G E N V A L U E   A N A L Y S I S   S U M M A R Y   (READ MODULE)

 ⋮
****************
  S L A V E   3
 ****************
 ⋮
E I G E N V A L U E   A N A L Y S I S   S U M M A R Y   (READ MODULE)

 ⋮
```

5.4 DMODES + serial optimization process for SOL 200

In the optimization procedures, several different types of analyses may be involved. If any eigensolution is required, the eigenvalue analysis may be performed in parallel with the computational methods FDMODES, GDMODES, and HDMODES used as in SOL 103.

In each iteration of the optimization procedure, the optimization algorithm runs in serial on each processor except for the eigenvalue analysis. Whenever the eigenvalue analysis is performed in parallel, the collected and merged eigensolutions from the master processors are broadcast to all processors. This approach results in the same optimization results for every processor, because all processors proceed with the optimization using the same eigensolutions inside the optimization loop. The DMP task of optimization procedure is described in Figure 5.4. The method selection of DMP normal mode analysis is similar to that of SOL 103, which is introduced in Figure 4.6.

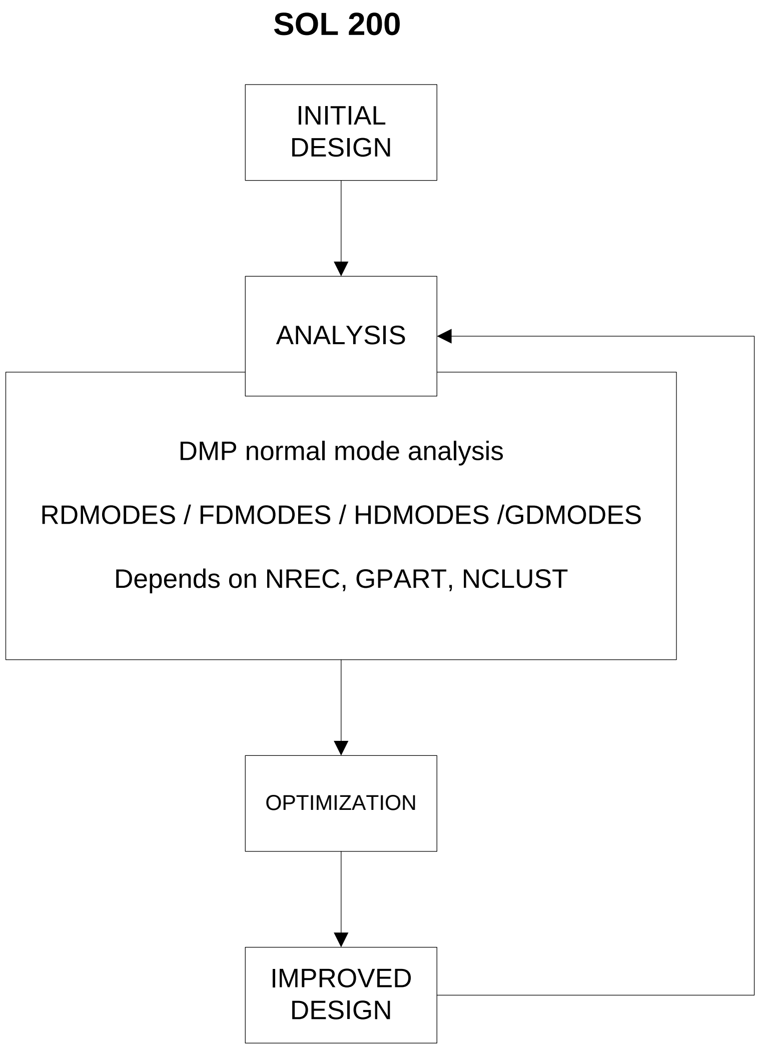


Figure 5.4 DMP design optimization (SOL 200)

Example: SOL 200 (DMODES + Serial optimization process)

```
nastran example dmp=4 (and/or nclust=2/numseq=4/nrec=n)
```

The following .f06 is a typical output format in SOL 200 with DMP, in which all processors print the same output.

```
⋮
i-th  I T E R A T I O N
⋮
$-----------------------------------------------------------------------------------
$   run FDMODES, GDMODES, or HDMODES, and collect the local eigensolutions
$   from the slave processors
$-----------------------------------------------------------------------------------

E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)

⋮

O P T I M I Z A T I O N   O U T P U T
⋮

j-th  I T E R A T I O N
⋮
****************
  S L A V E   1
 ****************
⋮
i-th  I T E R A T I O N
⋮
$-----------------------------------------------------------------------------------
$    run FDMODES, GDMODES, or HDMODES, and have the same eigensolutions
$    as the master processor
$-----------------------------------------------------------------------------------

E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)

⋮

O P T I M I Z A T I O N   O U T P U T
⋮


j-th  I T E R A T I O N
⋮
****************
  S L A V E   2
 ****************
⋮
i-th  I T E R A T I O N
⋮
$-----------------------------------------------------------------------------------
$    run FDMODES, GDMODES, or HDMODES, and have the same eigensolutions
$    as the master processor
$-----------------------------------------------------------------------------------

E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)

⋮

O P T I M I Z A T I O N   O U T P U T
⋮
```

```
j-th  I T E R A T I O N
 ⋮
****************
  S L A V E   3
 ****************
 ⋮
i-th  I T E R A T I O N
 ⋮
$-----------------------------------------------------------------------------------
$    run FDMODES, GDMODES, or HDMODES, and have the same eigensolutions
$    as the master processor
$-----------------------------------------------------------------------------------

E I G E N V A L U E  A N A L Y S I S  S U M M A R Y   (READ MODULE)

 ⋮

O P T I M I Z A T I O N  O U T P U T
 ⋮
j-th  I T E R A T I O N

 ⋮
```

Chapter 6 – Performance Study

6.1 Performance of Parallel Processing

The performance of a parallel processing is rated by its speedup or efficiency. The speedup S_p is defined as:

$$S_p = \frac{T_s}{T_p} = E * p$$

where *Ts* is the time to run a serial job, and *Tp* is the time it takes to run the same job with *p* processors.
The efficiency *E* is defined as:

$$E = \frac{T_s}{(p * T_p)}$$

For better performance, it is helpful to minimize the communication overhead by using high speed network switches. The ideal is to have high bandwidth and low latency.

6.2 Industrial Case Study 1

A trimmed car body FE model with SOL 103 is used as a case study to analyze the technologies presented in NX Nastran. This type of car model has all major components of the car, such as wheels, engine, etc. incorporated. Tables 6-1 and 6-2 present the details for this finite element model.

Table 6-1 Model statistics of trimmed car body FE model

Number of Nodes	Number of shell elements	Number of solid elements	Number of rigid elements
380,007	361,249	3,762	9,056

Table 6-2 The size of sets for the trimmed car body FE model

The size of set			
g-set	n-set	f-set	a-set
2,280,042	2,223,139	2,223,109	1,937,282

Table 6-3 demonstrates the effect of the automated geometric domain decomposition on this model by showing the number of interior and boundary nodes of the partitions. Several observations can be made. The interior range size depends on the quality of the automated partitioning. The boundary size increases with the number of partitions. Finally, the boundary size is at least two orders of magnitude smaller than the interior size, which is important to the computational efficiency.

Table 6-3 Results of geometry domain decomposition

Number of Partitions	Maximum Interior	Minimum Interior	Maximum Boundary	Minimum Boundary
2	197,460	182,199	354	354
4	104,788	90,791	572	418
8	54,496	40,067	518	450

The automated frequency domain decomposition results are shown in Table 6-4. There are 840 modes in the frequency range of interest.

Table 6-4. Frequency domain decomposition statistics

Number of segments	Minimum number of modes among segments	Maximum number of modes among segments
2	380	460
4	193	252
6	133	165
7	105	141
8	92	119

The automated geometric domain partitioning techniques usually provide only even, and preferably binary, numbered domains. This is because these techniques are primarily based on binary graph partitioning. This is not a restriction, as shared memory workstations tend to have an even number of processors. On the other hand, in the frequency domain decomposition, odd numbers of segments are also allowed. This technology is insensitive to that issue and enables the use of odd-numbered workstations via the hierarchic technology in a workstation cluster environment.
The analysis was executed on a cluster of eight workstations, each containing eight processors with a 1.5 GHz clock cycle. The cluster had a one gigabyte Ethernet network connection. The option gpart=1 is used.

Table 6-5. Execution times on workstation cluster with HDMODES

Number of processors	Number of partitions	Number of segments	I/O (GB)	Elapsed time[min:sec]	Speedup
1	1	1	1,028.4	528:58	1.00
4	2	2	431.4	167:15	3.13
8	4	2	266.9	83:41	6.26
16	8	2	191.1	45:16	11.57
32	8	4	98.3	34:07	15.35
48	8	6	77.8	27:14	19.23
56	8	7	67.1	24:41	21.22
64	8	8	61.4	27:00	19.40

The task of finding the natural frequencies and mode shapes of such a model is an enormous one. It is an overnight job with more than a terabyte of I/O operations. The execution on a single processor is impractical considering the work environment and time schedule at automobile companies.

Fig. 6.1 and Fig. 6.2 show the elapsed time and disk I/O of HDMODES with 8 geometry partitions for different numbers of frequency segments. With 56 processors, 8 geometry partitions in each workstation and 7 frequency segments across workstations are used. The elapsed time for 32 processors is already a practical execution. The efficiency above decreases, but the speedup is still increasing. It peaks at 56 processors, although a wider frequency range for this model may extend that peak to 64 or higher. It means that 7 frequency segments with 8 geometry partitioning for each frequency segment is the most suitable partitioning for this test FE model in this workstation cluster environment.

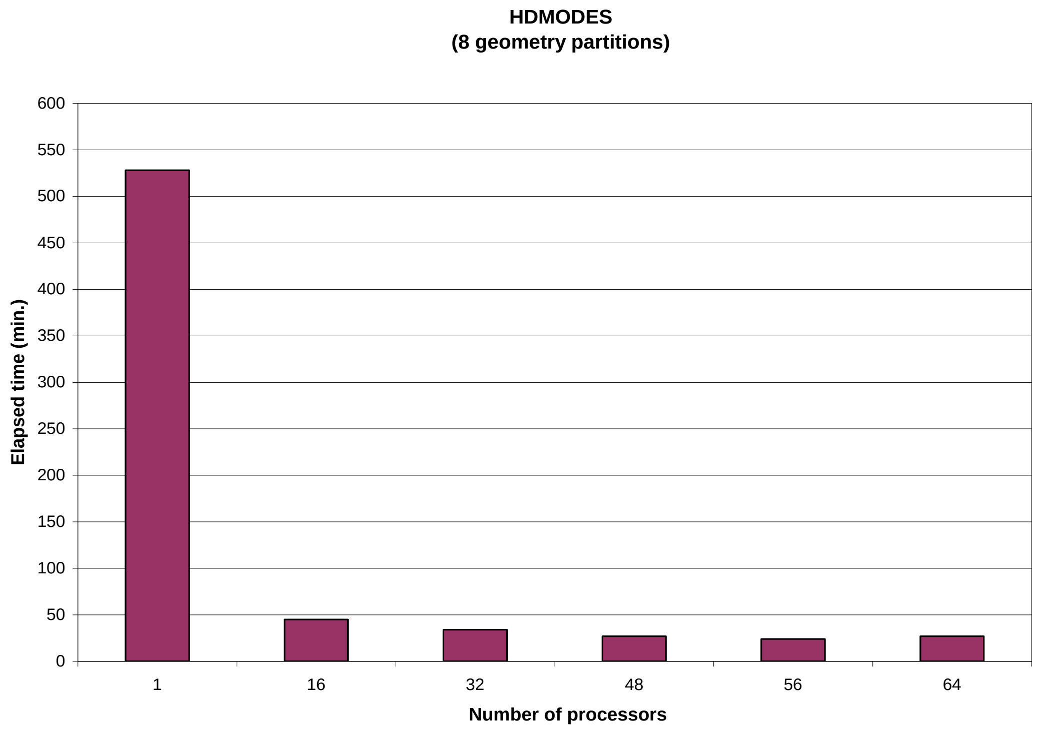


Fig. 6.1 Elapsed time of HDMODES with 8 geometry partitions for different numbers of frequency segment.

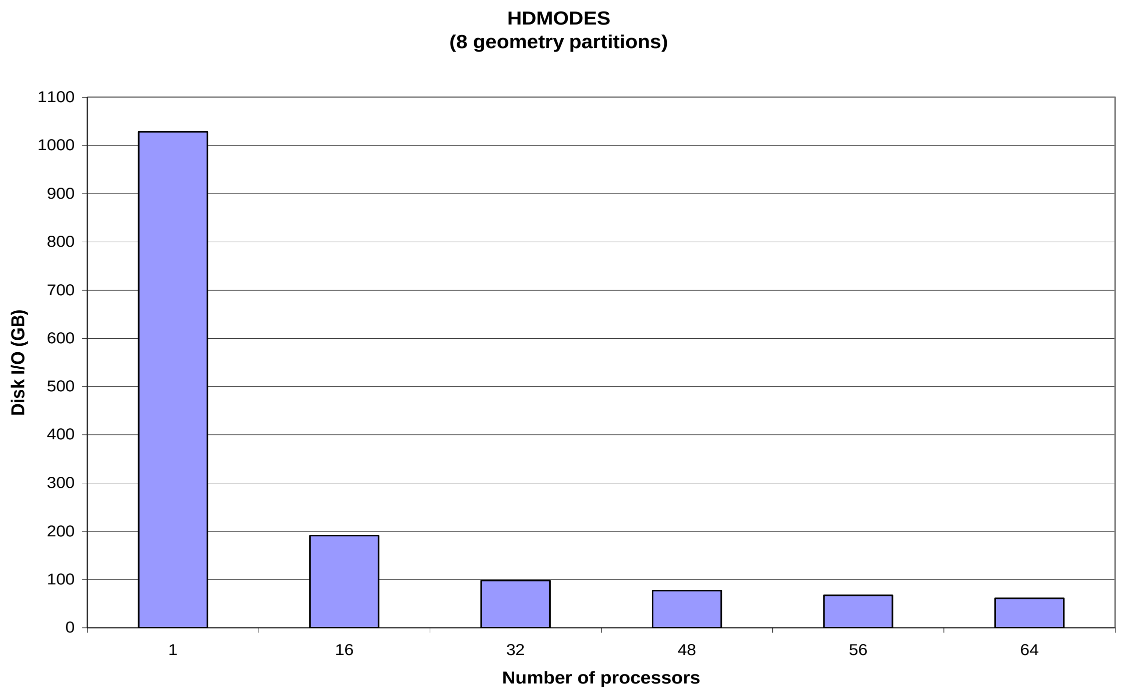


Fig. 6.2 Disk I/O of HDMODES with 8 geometry partitions for different numbers of frequency segment.

6.3 Industrial Case Study 2

A trimmed car body FE model with SOL 103 is used as a case study to analyze the performance of RDMODES in NX Nastran. This model has g-size around 20 million and f-size about 10 million. The number of grid points is around 3.6 million, and the number of ctetra elements is about 2.3 million.

The example was run through RDMODES with rdsparse (sparse eigenvector recovery, new in NX Nastran 7.0) turned on and nrec=256.

Scalability – elapsed time vs. number of processors

It computed modes up to 10,000 Hz. The following table and graph will show the summary of the performance from one processor to 64 processors.

Table 6-6. Execution times on workstation cluster with RDMODES

Number of processors	Elapsed time (min:sec)	I/O (G-bytes)	CPU (seconds)	speedup
1	4370:52	9110.7G	186277	1.00
2	2548:23	5136.2G	110519.1	1.78
4	1402:05	2711.5G	58795.8	3.12
8	1019:13	1431.5G	29086.3	4.29
16	678:09	945.6G	19272.7	6.45
32	505:36	637.8G	10731.6	8.64
64	354:53	526.3G	8920.1	12.32

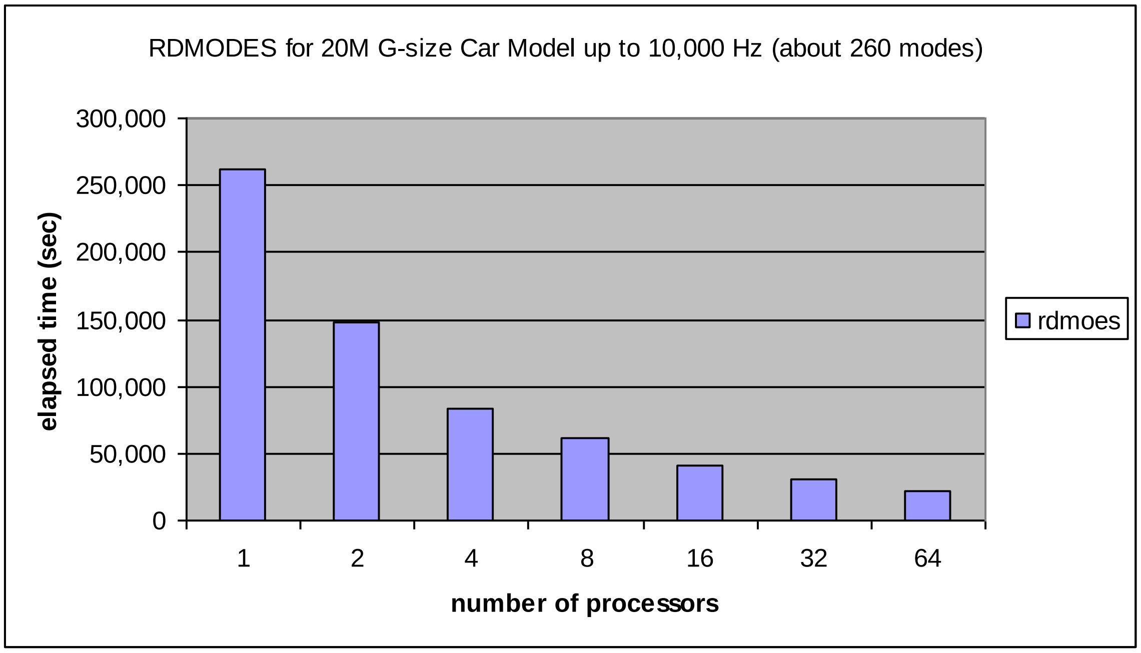


Fig. 6.3 Elapsed time of RDMODES with 256 geometry partitions for the different number of processors.

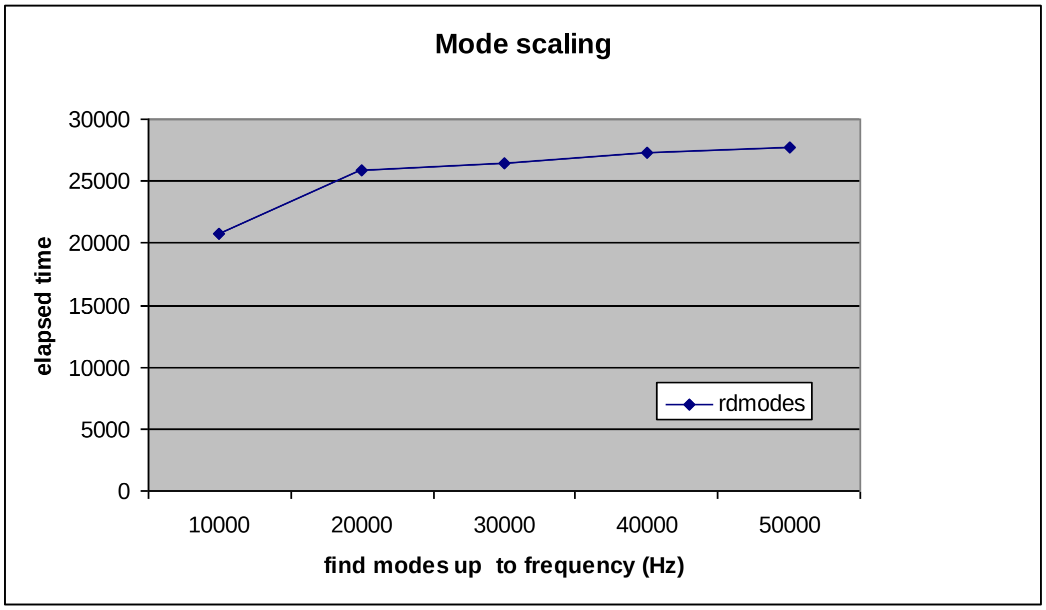
The analysis was also executed on a Linux cluster of 64 nodes; each processor is 1.8 GHz clock cycle. The cluster had a one gigabyte Ethernet network connection.
Due to the large dimension of this model, the task of finding the natural frequencies and mode shapes of such a model is more enormous. The elapsed time is saved significantly through RDMODES with more processors.

Table 6-6 and Figure 6.3 showed that, when 64 processors are used, the computation is speedup by 12. The elapsed time is around 6 hours with 64 processors, while the time of one processor is 73 hours.

Scalability – elapsed time vs. number of modes

The following table and graph show multi-level RDMODES runs (dmp=64, nrec=256) with respect to frequency range up to 10000, 20000, 30000, 40000 and 50000 Hz. It took about 110 extra minutes and found about 2900 more modes by increasing frequency range from 10000 to 50000. It clearly demonstrates that the RDMODES is capable for computing large number of modes.

Modes below (Hz)	RDMODES	
	Elapsed time	Number of modes
10000	354:53	295
20000	430:47	764
30000	441:20	1453
40000	455:09	2276
50000	462:04	3255



Chapter 7 - Installation and Configuration of DMP

7.1 Overview

NX Nastran offers the ability to run certain solution sequences in parallel using the Message Passing Interface (MPI), an industry-wide standard library for C and Fortran message-passing programs. MPI programs can be run on SMP computers, NUMA computers, distributed computers, and any collection of computers supported by the MPI package.

Note: Further information on MPI can be obtain online at

http://www.mpi-forum.org

MPI is included with the installation of NX Nastran.

Special Considerations

To install NX Nastran for Distributed Memory Parallel (DMP) operations, you must select one of the following three installation schemes if you want to use more than one host in a single NX Nastran job:

- Install NX Nastran on a filesystem that is global to every host. This provides the easiest installation and system administration, but may present network load issues when the NX Nastran is started and the delivery databases are being read.
- Install NX Nastran on every host on host-private filesystems. This is harder to install and administer, but reduces the network load when NX Nastran is started.
- A combination of the above.

Note: In all cases, the nastran command must have the same pathname, or be in the default PATH of every host that will run a DMP job. Recall that your ".profile" and ".login" files are not used for rcp(1) and rsh(1) operations.

7.2 Requirements

Platform	MPI
X86_64 Linux	Intel MPI is included with the NX Nastran installation, and is automatically invoked when a DMP job is executed on Linux. Intel MPI has specific requirements, for example, Python must be installed. These requirements are documented in the Intel MPI release notes found at: https://software.intel.com/en-us/articles/intel-mpi-library-documentation

Windows-64	Intel MPI is included with the NX Nastran installation. See the Windows Single Host Instructions.

In the descriptions that follow, the "local" node is the computer you issue the nastran command on, the "master" node is the first computer named by the "hosts" keyword, and the "slave" nodes are the remaining systems listed in the "hosts" list.

The following are some general requirements for running NX Nastran DMP jobs:

- NX Nastran must be properly installed on all the hosts listed by the "hosts" keyword.
- On Linux, either rsh or ssh can be used as a remote command. Secure Shell (ssh) is supported on Linux provided that:
  a. The environment variable MPI_REMSH is set to ssh.
  b. The argument s.rsh=ssh is included on the nastran command line.
  c. The argument s.rcp=scp is included on the nastran command line.
- You must have access to each system you want to access in a distributed job. For example, when using rsh, you can test this with the command:
  ```
  rsh <node>  [-1 <username>] date
  ```
  where <node> is the name of the node and <username> is an alternate username on the remote system if your current username is not valid. For example:
  ```
  rsh node1 date
  ```
  The output from the above command should be in a single line containing the current date on node1 in a format similar to
  ```
  Thu Jul 17 13:06:49 EST 2003
  ```
  If any other output is present, you should determine the source of the output and correct the problem. If you cannot eliminate the output, you will not be able to use the distributed execution capabilities of the nastran command.
- You must have "remote execution" privileges on all the hosts listed by the "hosts" keyword. That is, a password must not be required to execute a remote copy (rcp) or remote shell (rsh or remsh) command. See your system administrator for information on this.
- The input data file must be accessible on the local host.
- INCLUDE files must be local-to, or visible-from, each host.
- All default output files, i.e., those without ASSIGN statements, will be written to a directory accessible to the local host.
- The scratch directory can be a global or local file system. Your scratch directory should be local to each host, i.e., you specify per-host "sdirectory" values.
- The pathname of the nastran command must be the same on all hosts, or on the default PATH of each host, used in the analysis.
- If you execute a restart, you must specify the identical values for "dmparallel" and "hosts" as were used on the cold start.
- In a restart, i.e., a job that uses an existing database, the DBSets must be local-to, or visible-from, the remote system.

Note: Recall that remote executions do not run a "login" shell. That is, your ".profile" or ".login" script is not executed.

When running a DMP job, nastran keywords are processed on both the local and master/slave systems. Keywords that control the job's output and interaction with you are processed on the local system. These are:

Keyword	Description
append	Requests the .f06, .f04, and .log files to be concatenated.
dmparallel (or dmp)	Specifies the number of tasks for a Distributed Memory Parallel (DMP) analysis. This value may only be set on the command line.
gpart	Selects the geometry partitioning option for a hierarchic dmp (HDMP) solution.
hostovercommit	Requests more tasks per host than CPUs.
hosts	Specifies list of hosts to use. Separate hosts with the PATH separator, i.e, ;.
mergeresults	Specifies the results from each DMP task are to be merged into the standard files from the master host.
nclust	Specifies the number of frequency segments for a hierarchic dmp (HDMP) solution.
ncmd	Specifies an alternate notification command
notify	Requests notification when the job completes.
old	Specifies versioning or deletion of previously existing output files.
oldtypes	Specifies additional user file types to be versioned or deleted.
out	Specifies an alternate output file prefix.
rcmd	Specifies the nastran command path on the master/slave systems.
scratch	Specifies the database DBSets are to be deleted at job completion.
sdirectory	Specifies each per-host directory to contain NX Nastran temporary files. Separate directories with the PATH separator.
slaveout	Specifies the .f04 and .f06 files from the slave tasks are to be appended to the .f04 and .f06 files of the master task.
xmonitor	Requests XMONITOR to monitor the master task's progress.

The "sdirectory" keyword is special, as the command line, RC files on the current host, and RC files on the each master and slave host will all be considered when establishing a scratch directory. All remaining keywords are only scanned on the master and slave systems.

Once "dmparallel=number" is processed, the following processing takes place:

1. Process the RC files on the local system if the "version" keyword has been defined in the command initialization file or the command line.
2. Process the RC file specified by the "rcf" keyword if it was defined on the command line.
3. Determine the full pathname of the input file so that its visibility from the master and each slave host can be tested.
4. Create a "touch" file in the specified output file so that its visibility from the master and each slave host can be tested.

5. If the “dmpdeny” utility, i.e., install_dir/nxnr/arch/dmpdeny, exists and is executable, run it, and save its output.
6. If the “dmpaccept” utility, i.e., install_dir/nxnr/arch/ dmpaccept, exists and is executable, run it, and save its output.
7. Ensure “scratch=no” was set if the “dbs” keyword was set.
8. Determine every possible pairing of host and sdirectory by scanning each list in a round-robin order. That is, the first host is paired with the first sdirectory, the second host with the second sdirectory, and so on.
9. Execute the following steps for each host-sdirectory pair determined above until host-sdirectory pairs have been assigned to each of the tasks requested by the “dmparallel” keyword or no more host-sdirectory pairs are available. Steps 9a. through 9f. are executed only once per host-sdirectory pair.
    a) Verify that host exists and you are able to run a command on that system.
    b) If the “rcmd” keyword was specified, attempt to execute that command on host, display an error and cancel the job if it fails. Otherwise attempt to execute the pathname of the current nastran command on host. If it fails, attempt to execute the basename of the current nastran command on host. Display an error and cancel the job if both checks fail.
    c) Run the remote nastran command identified in the previous step to determine: if the input data file is visible; if the “touch” file is visible, if the “sdirectory” (if identified on the local system) exists; if the “dbs” directory (if identified on the local system) exists; the “sdirectory” value in the RC files defined on host; and finally the numeric format of host.
    d) Drop this host-sdirectory pair from further consideration if a scratch directory was identified on the command line or in a local RC file, but does not exist on host.
    e) Display an error and cancel the job if the numeric format of host differs from the numeric format of the local host.
    f) Display an error and cancel the job if the directory specified by a “dbs” keyword on the command line or in a local RC file does not exist on host.
    g) Assign the current host-sdirectory pair to the next task; save the per-host visibility flags, “rcmd”, and “sdirectory” values.
10. Display an error and cancel the job if one or more of the tasks requested by the “dmparallel” keyword have not been assigned.
11. Delete the “touch” file created above.
12. The remaining steps are done in a background process (possibly some time later) if “batch=yes” or “after” was specified.
    a) Copy the input data file to the scratch directory of any host that could not see the input data file.
    b) Set “out” to the host-specific scratch directory value of every host that could not see the output directory.

c) Copy the remaining keywords on the command line that were not processed, to a local RC file in the scratch directory on the remote node.
d) Run the DMP job using the system's MPI startup command. Note that each task will write its files to task-specific names.
e) Process the "old" and "oldtypes" keywords on the local node.
f) Copy the output files (.f04, .f06, .log, .ndb, .pch, .plt) from the master task to the directory specified by the "output" keyword and delete the files from the master node if it could not see the output directory.
g) Process the "append" keyword on the local node.
h) Process the "notify" keyword on the local node.

Once the job has completed, the .f06, .f04, .log, .ndb, .op2, .plt, .pch, and .xdb files from the master task will be present as if the job were run locally.

Note: No attempt is made to copy DBSet files between the local and master/slave systems. If this is required, you must handle this yourself and set the "dbs" keyword appropriately.

7.3 Windows Single Host Instructions

The following instructions describe how to install Windows DMP on a single host which includes multiple cores.

Step 1: Before you can follow the remaining steps, an administrator will need to perform the following tasks.

- You must have a login directory that you own.
- You must have full permissions to all working directories and scratch directories.
- Your working directory must be shared, for example, `D:\workdir` shared as `\\host\workdir`.

Step 2: Install NX Nastran to the Windows host

- Install NX Nastran to a directory with no spaces in the path. For example, do not install to the default path of `C:\Program Files`. Spaces in the path will prevent environment variables from being properly interpreted.

Step 3: Set up the Intel MPI Windows Service

- Login as the local administrator and open a DOS shell.
- Go to the bin directory:
  cd /d …*installation_path*\nxnr\em64tnt\impi\bin
- Enter the following:
  `hydra_service.exe -install`

Step 4: Unset **PLATFORM_MPI**

- The environment variable **PLATFORM_MPI** was required in previous releases to run DMP. It must be unset if it is still defined on your system. Enter the following to check if it is defined:
  ```
  echo %PLATFORM_MPI%
  ```
  %PLATFORM_MPI% will return if it is undefined.
  If it is defined, a file system path will appear.
  Enter the following if you need to unset the variable:
  ```
  set PLATFORM_MPI=
  ```

Step 5: Each user must set up password-less MPI in a DOS shell

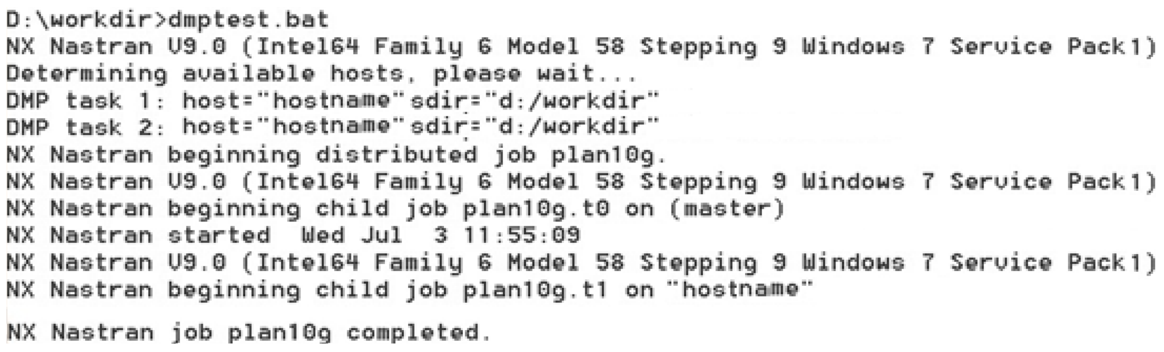
- Go to the bin directory:
  ```
  cd /d …installation_path\nxnr\em64tnt\impi\bin
  ```
- Enter the following:
  ```
  mpiexec.hydra.exe –register
  ```
- Follow the prompts to cache your password.
- Repeat these steps whenever you change your password.

Step 6: Define MPI_ROOT and update PATH

- Define the MPI_ROOT environment variable:
  ```
  set MPI_ROOT=installation_path\nxnr\em64tnt\impi
  ```
- Include the following ‘bin’ directory in your PATH variable:
  ```
  set PATH=installation_path\nxnr\em64tnt\impi\bin;%PATH%
  ```

Step 7: Test Windows DMP with a .BAT script

- Use a text editor to create the following .BAT file, replacing hostname with the actual host name. The example assumes a ‘workdir’ and enabled sharing for this folder, and an NX Nastran installation location of D:\NXN*r*. Note the use of quoted, forward slashes for the shared directory.
  Name the file dmptest.BAT:
  ```
  @ECHO OFF
  set NXN_BASE=D:\NXNr
  %NXN_BASE%\bin\nastran.exe ^
  "// hostname /workdir/plan10g.dat" ^
  out="// hostname /workdir" ^
  mem=130mw dmp=2 hosts= hostname ^
  sdir=D:\Scratch slaveout=yes scr=yes
  ```
- Open a DOS shell, and change directories to ‘workdir’:
  ```
  cd \\hostname\workdir
  ```
- Copy the file plan10g.dat from the tpl directory:
  ```
  installation_path\nxnr\nast\tpl\plan10g.dat
  ```
  to `//hostname/workdir/`
- Execute the .BAT file:
  ```
  dmptest.bat
  ```
- The results should look similar to the following:



```
D:\workdir>dmptest.bat
NX Nastran V9.0 (Intel64 Family 6 Model 58 Stepping 9 Windows 7 Service Pack1)
Determining available hosts, please wait...
DMP task 1: host="hostname" sdir="d:/workdir"
DMP task 2: host="hostname" sdir="d:/workdir"
NX Nastran beginning distributed job plan10g.
NX Nastran V9.0 (Intel64 Family 6 Model 58 Stepping 9 Windows 7 Service Pack1)
NX Nastran beginning child job plan10g.t0 on (master)
NX Nastran started  Wed Jul  3 11:55:09
NX Nastran V9.0 (Intel64 Family 6 Model 58 Stepping 9 Windows 7 Service Pack1)
NX Nastran beginning child job plan10g.t1 on "hostname"

NX Nastran job plan10g completed.
```

7.4 Windows Multiple Host Instructions (True Cluster)

The following instructions describe how to install Windows DMP across multiple hosts (in a cluster). See the Windows Single Host Instructions to install and run Windows DMP on a single host which includes multiple cores.

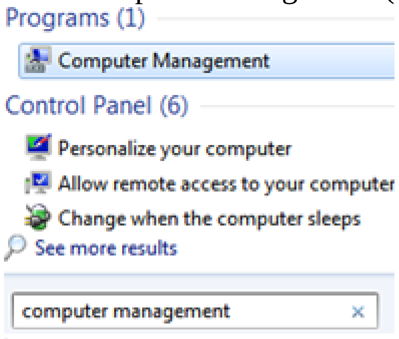
General Requirements

- NX Nastran must be installed in the same location on each node and must be accessible to each user.
- Each user must have full permissions to all working directories and scratch directories.
- All working directories must be shared to all nodes.
- Only Server 2008 and Server 2012 are supported in the true cluster mode.
- DMP Jobs must be launched from a DOS shell.

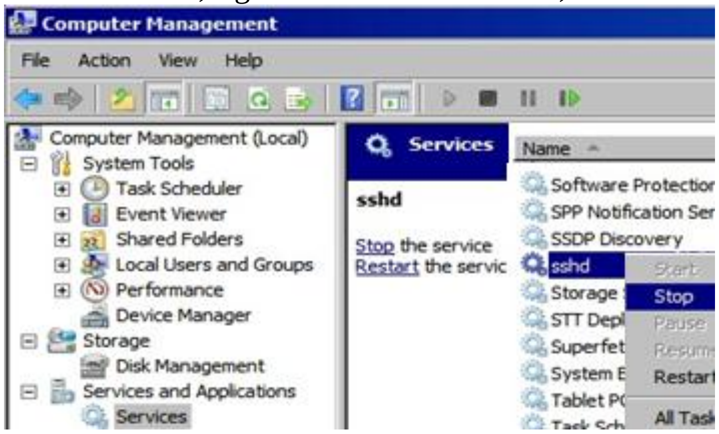
Step 1: SUA Removal Instructions
Since June 1, 2013, Microsoft has removed support for SUA and recommends that Cygwin be installed in its place:
http://technet.microsoft.com/library/hh831568.aspx
If SUA is installed on your cluster, it must be removed before installing Cygwin 64-bit. You can follow the following SUA removal instructions. Note that the SUA removal instructions must be repeated on each node of the cluster.
If you do not have SUA installed, skip forward to the Cygwin 64-bit installation instructions in Step 2.

A. Stop the SSHD service
   - Login as Administrator

- Start Computer Management (Start, Search, “Computer Management”)


Programs (1)
Computer Management
Control Panel (6)
Personalize your computer
Allow remote access to your computer
Change when the computer sleeps
See more results
computer management

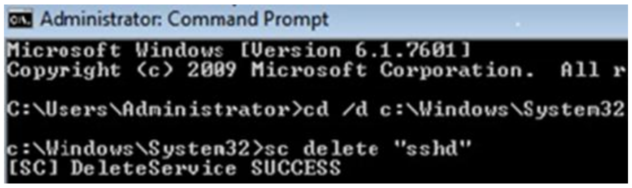
- Select Services, right-click the sshd service, and select stop



- When the service is stopped, close the Manage window

B. Remove the SSHD service
- Login as Administrator
- Open a Command Prompt and cd to `C:\Windows\System32`
- Remove the SSHD service with the command

```
        sc delete "sshd"
```

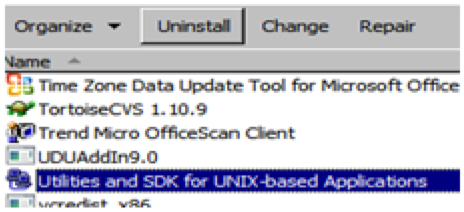


```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All r

C:\Users\Administrator>cd /d c:\Windows\System32

c:\Windows\System32>sc delete "sshd"
[SC] DeleteService SUCCESS
```

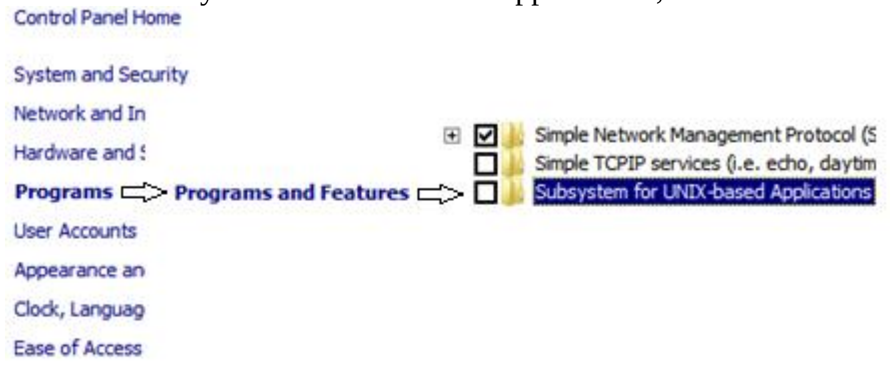
C. Remove the SUA SDK
- Login as Administrator
- Open Control Panel, select Uninstall a program
- Select “Utilities and SDK for UNIX-based Applications”, and select Uninstall



Time Zone Data Update Tool for Microsoft Office
TortoiseCVS 1.10.9
Trend Micro OfficeScan Client
UDUAddIn9.0
Utilities and SDK for UNIX-based Applications
vcredist_x86

D. Remove SUA as a Windows Feature

- Login as Administrator
- Open Control Panel, select Programs, then Programs and Features
- Select Turn Windows features on or off
- Un-check Subsystem for UNIX-based applications, and select OK

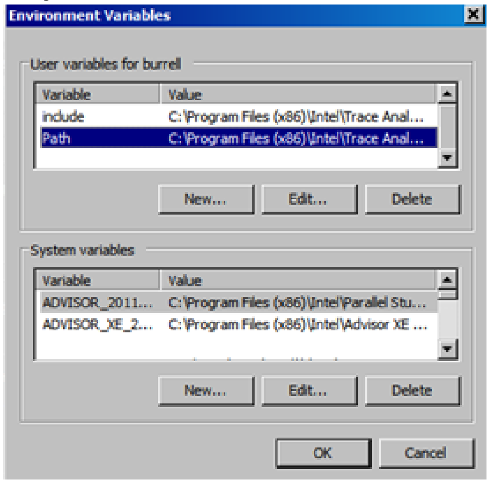


[✓] Simple Network Management Protocol (S
[ ] Simple TCPIP services (i.e. echo, daytim
[ ] Subsystem for UNIX-based Applications

- When asked to reboot the computer, select Restart Now.

E. Verify that SUA is no longer a part of any environment variable.

- After the previous re-boot, login as Administrator.
- Right click on the desktop Computer icon, select Properties, Advanced System Settings.
- Select Environment Variables and edit out any references to SUA for both User and System.



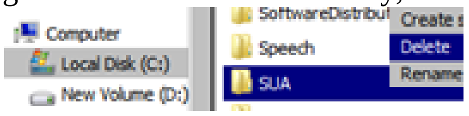
Variable	Value
include	C:\Program Files (x86)\Intel\Trace Anal...
Path	C:\Program Files (x86)\Intel\Trace Anal...

Variable	Value
ADVISOR_2011...	C:\Program Files (x86)\Intel\Parallel Stu...
ADVISOR_XE_2...	C:\Program Files (x86)\Intel\Advisor XE ...

F. Remove the SUA directory (optional)

- After the previous re-boot, login as Administrator

- Right click on the SUA directory, and select Delete



Step 2: Cygwin 64-bit Installation

Prerequisites

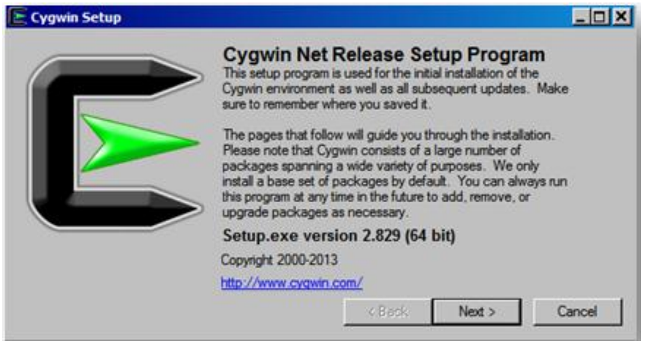
- Cygwin 64-bit is required for NX Nastran
- If a previous version of Cygwin is installed, it should be removed before installing Cygwin 64-bit.
- Instructions for removing previous versions of Cygwin can be found at http://cygwin.wikia.com/wiki/Uninstalling_Cygwin
- NX Nastran requires Cygwin-64 to be installed to the C:\ drive under C:\cygwin64.
- Cygwin-64 must be installed on each node of the cluster

Begin the Installation

A. Login as local Administrator

B. Go to http://cygwin.com/install.html, right-click "setup-x86_64.exe", choose save as target. Save to a local directory, such as `C:\Temp`.

C. Double-click on `C:\Temp\setup-x86_64.exe` to begin the installation. Select Next on the setup window:


Cygwin Setup

Cygwin Net Release Setup Program

This setup program is used for the initial installation of the Cygwin environment as well as all subsequent updates. Make sure to remember where you saved it.

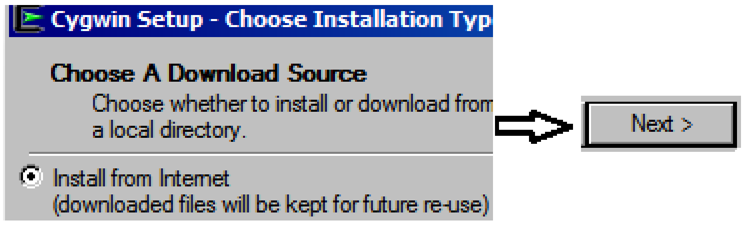
The pages that follow will guide you through the installation. Please note that Cygwin consists of a large number of packages spanning a wide variety of purposes. We only install a base set of packages by default. You can always run this program at any time in the future to add, remove, or upgrade packages as necessary.

Setup.exe version 2.829 (64 bit)

Copyright 2000-2013

http://www.cygwin.com/

D. Select "Install from Internet", then Next.

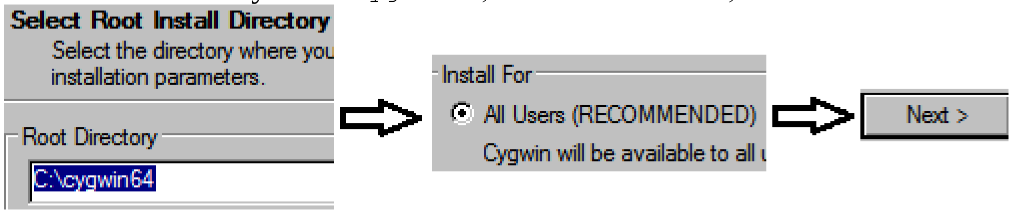

Cygwin Setup - Choose Installation Typ

Choose A Download Source

Choose whether to install or download from
a local directory.

[✓ Install from Internet]
(downloaded files will be kept for future re-use)

E. Make sure the installation directory is `C:\cygwin64`, select “All Users”, then Next.


Select Root Install Directory

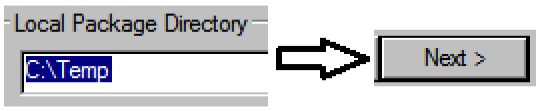
Select the directory where you
installation parameters.

Root Directory: C:\cygwin64

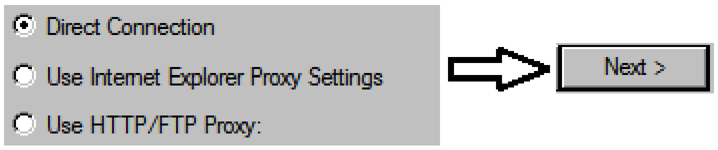
Install For

[✓ All Users (RECOMMENDED)]
Cygwin will be available to all u

F. Select a suitable scratch directory to hold installation files, such as `C:\Temp`. Select Next.


Local Package Directory: C:\Temp

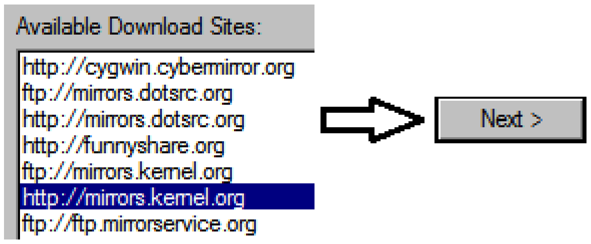
G. Select your internet connection, then Next.


[✓ Direct Connection]

Use Internet Explorer Proxy Settings

Use HTTP/FTP Proxy:

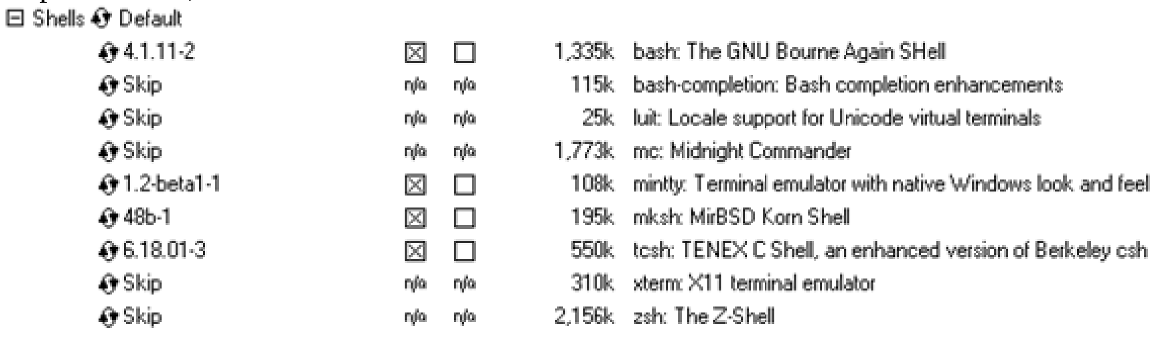
H. Choose a download site, for example http://mirrors.kernel.org. Select Next.


Available Download Sites:

- http://cygwin.cybermirror.org
- ftp://mirrors.dotsrc.org
- http://mirrors.dotsrc.org
- http://funnyshare.org
- ftp://mirrors.kernel.org
- [✓ http://mirrors.kernel.org]
- ftp://ftp.mirrorservice.org

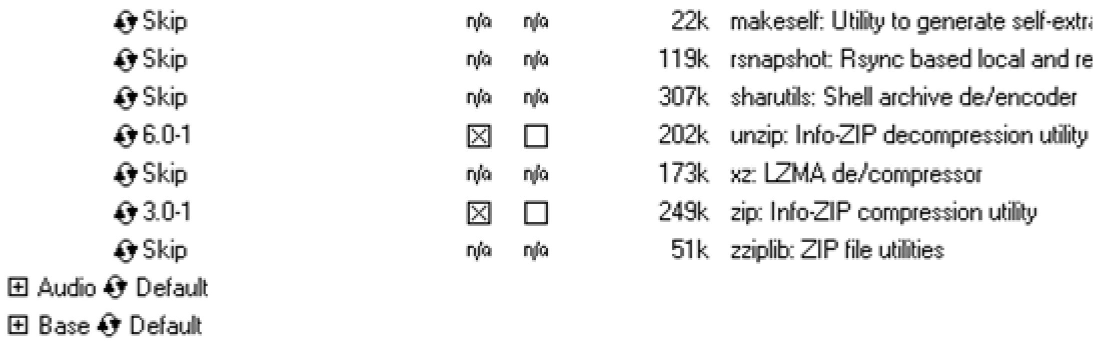
I. From Select Packages:

- Expand Shells, select tcsh and mksh.


⊟ Shells Default

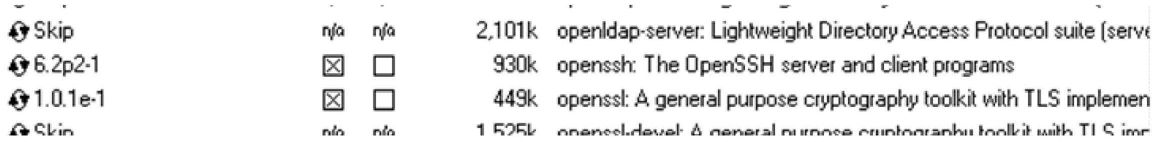
4.1.11-2	☒	☐	1,335k	bash: The GNU Bourne Again SHell
Skip	n/a	n/a	115k	bash-completion: Bash completion enhancements
Skip	n/a	n/a	25k	luit: Locale support for Unicode virtual terminals
Skip	n/a	n/a	1,773k	mc: Midnight Commander
1.2-beta1-1	☒	☐	108k	mintty: Terminal emulator with native Windows look and feel
48b-1	☒	☐	195k	mksh: MirBSD Korn Shell
6.18.01-3	☒	☐	550k	tcsh: TENEX C Shell, an enhanced version of Berkeley csh
Skip	n/a	n/a	310k	xterm: X11 terminal emulator
Skip	n/a	n/a	2,156k	zsh: The Z-Shell

- Expand Archive and select unzip and zip.



```
Skip      n/a  n/a   22k  makeself: Utility to generate self-extr
Skip      n/a  n/a  119k  rsnapshot: Rsync based local and re
Skip      n/a  n/a  307k  sharutils: Shell archive de/encoder
6.0-1     ☒    ☐    202k  unzip: Info-ZIP decompression utility
Skip      n/a  n/a  173k  xz: LZMA de/compressor
3.0-1     ☒    ☐    249k  zip: Info-ZIP compression utility
Skip      n/a  n/a   51k  zziplib: ZIP file utilities
⊞ Audio Default
⊞ Base Default
```

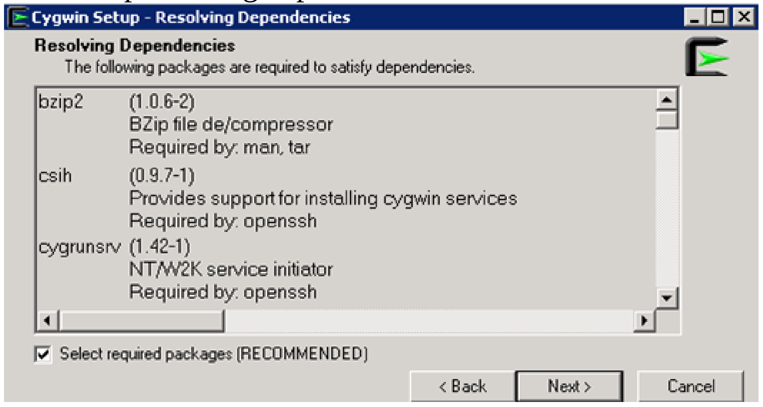
- Expand Net category and select openssh and openssl.



```
Skip      n/a  n/a  2,101k  openldap-server: Lightweight Directory Access Protocol suite (serve
6.2p2-1   ☒    ☐      930k  openssh: The OpenSSH server and client programs
1.0.1e-1  ☒    ☐      449k  openssl: A general purpose cryptography toolkit with TLS implemen
Skip      n/a  n/a  1,525k  openssl-devel: A general purpose cryptography toolkit with TLS imp
```

Additional packages can be selected later after the installation. Select Next to continue.

J. A window will open listing dependencies. Select Next to resolve dependencies.



Cygwin Setup - Resolving Dependencies

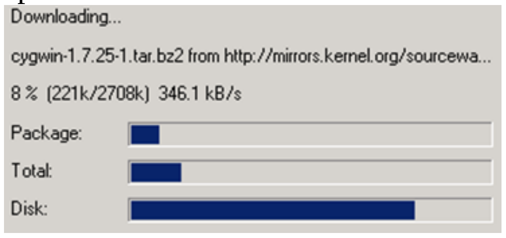
Resolving Dependencies

The following packages are required to satisfy dependencies.

```
bzip2      (1.0.6-2)
           BZip file de/compressor
           Required by: man, tar
csih       (0.9.7-1)
           Provides support for installing cygwin services
           Required by: openssh
cygrunsrv  (1.42-1)
           NT/W2K service initiator
           Required by: openssh
```

☑ Select required packages (RECOMMENDED)

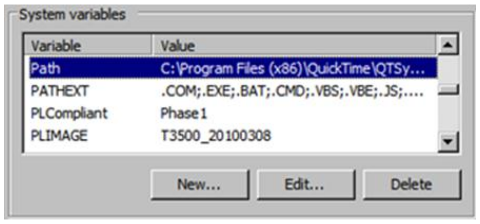
K. A window will open to show the installation progress. Select Finish after the installation completes.



```
Downloading...
cygwin-1.7.25-1.tar.bz2 from http://mirrors.kernel.org/sourcewa...
8 %  (221k/2708k)  346.1 kB/s
Package:
Total:
Disk:
```

Step 3: Cygwin 64-bit Configuration

A. Right-click the Computer icon, select Advanced Settings, then Environment Variables. Add `C:\cygwin64\bin` to the global PATH in the System Variables.



Variable	Value
Path	C:\Program Files (x86)\QuickTime\QTSy...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PLCompliant	Phase1
PLIMAGE	T3500_20100308

B. Administrator setup of SSH service

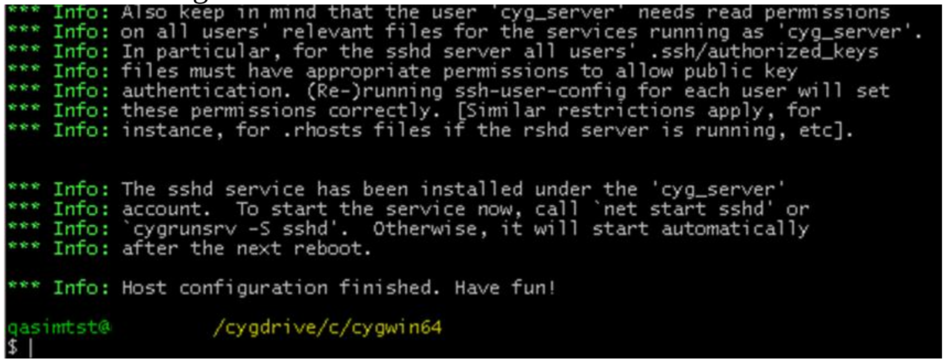
Login as Administrator, open a Cygwin64 terminal, and issue the command:

```
ssh-host-config
```

You will be prompted to answer several questions:

Questions	**Answers**
Should privilege separation be used?	Yes
Should this script create a new local account ‘sshd’?	Yes
Do you want to install sshd as a service?	Yes
Enter the value of CYGWIN for the daemon.	ntsec
Do you want to use a different name for the SSH service?	This is optional, but it is simplest to enter No.

Once the setup of the SSH service is complete, you should see something similar to the following:



```
*** Info: Also keep in mind that the user 'cyg_server' needs read permissions
*** Info: on all users' relevant files for the services running as 'cyg_server'.
*** Info: In particular, for the sshd server all users' .ssh/authorized_keys
*** Info: files must have appropriate permissions to allow public key
*** Info: authentication. (Re-)running ssh-user-config for each user will set
*** Info: these permissions correctly. [Similar restrictions apply, for
*** Info: instance, for .rhosts files if the rshd server is running, etc].


*** Info: The sshd service has been installed under the 'cyg_server'
*** Info: account.  To start the service now, call `net start sshd' or
*** Info: `cygrunsrv -S sshd'.  Otherwise, it will start automatically
*** Info: after the next reboot.

*** Info: Host configuration finished. Have fun!

qasimtst@         /cygdrive/c/cygwin64
$ |
```

Note: If you plan to use a Management Agent, then follow the instructions given here: http://docs.oracle.com/cd/E24628_01/install.121/e22624/preinstall_req_cygwin_ssh.htm

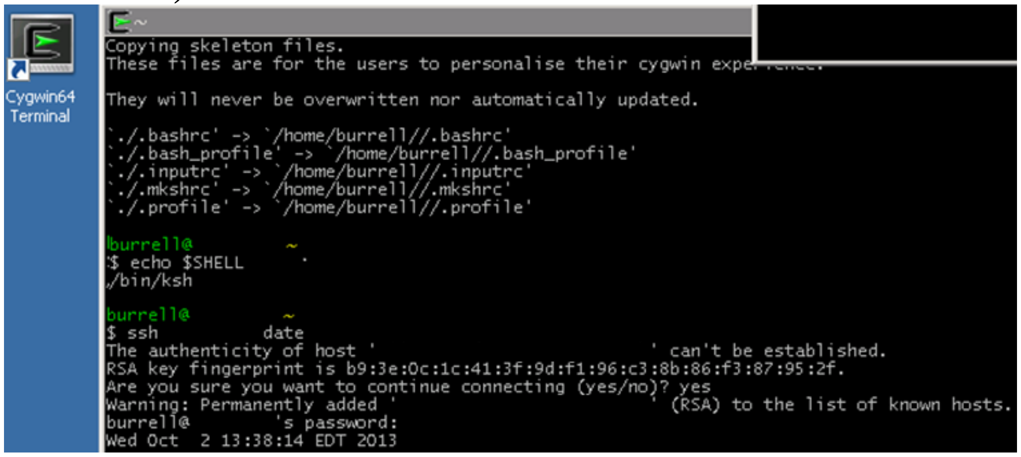
C. Startup the SSH service from a Cygwin64 terminal with:

```
cygrunsrv -S sshd
```

Once the service starts, the command prompt should return without error. The service will also automatically start after each reboot. You can also verify that the service is running by opening Control Panel, System and Security, and Selecting Administrative tools, and clicking on Services.

D. Logout from the local Administrator account, then login as one of the cluster users. Double-click on the Cygwin64 Terminal icon from the desktop. Note that a new home directory is created as each user logs in.

Verify that your shell shows correctly with
'`echo $SHELL`' (as shown below).
Also verify that ssh is working, and that you can run ssh on the localhost (as shown below).



```
Copying skeleton files.
These files are for the users to personalise their cygwin expe

They will never be overwritten nor automatically updated.

`./.bashrc' -> `/home/burrell//.bashrc'
`./.bash_profile' -> `/home/burrell//.bash_profile'
`./.inputrc' -> `/home/burrell//.inputrc'
`./.mkshrc' -> `/home/burrell//.mkshrc'
`./.profile' -> `/home/burrell//.profile'

burrell@          ~
$ echo $SHELL
/bin/ksh

burrell@          ~
$ ssh          date
The authenticity of host '                              ' can't be established.
RSA key fingerprint is b9:3e:0c:1c:41:3f:9d:f1:96:c3:8b:86:f3:87:95:2f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '                              ' (RSA) to the list of known hosts.
burrell@        's password:
Wed Oct  2 13:38:14 EDT 2013
```

Note: The Administrator can edit /etc/passwd to change the shell assigned to each user, after each user has logged in and their /home directory has been created.

E. Final Cygwin 64-bit Configuration Steps
   - For each user of the cluster, repeat the previous step until all users have a home directory on the local node, and each can run ssh on the local node. (Instructions to setup password-less ssh are provided later.)
   - If necessary, the local Administrator can open a Cygwin64 terminal and edit /etc/passwd to define the appropriate shell for each user.
   - Each user's login directory must exist on all nodes with normal permissions to the user
   - Each user's shell should be set (for example, to /bin/ksh)
   - Repeat all of the above installation instructions for Cygwin 64-bit on each additional node in the cluster.

After the installation and configuration of Cygwin 64-bit on each node of the cluster, the final step below shows how to setup password-less ssh for each user, install the latest version of NX Nastran, and configure MPI on the cluster for each user.

Step 4: Remaining Installation Tasks for NX Nastran DMP

The remaining tasks to be completed in Step 4 are:

A. Set up password-less SSH with Cygwin 64-bit for each user
B. Install the latest release of NX Nastran or modify an existing release
C. Set up Intel MPI Windows services
D. Set up password-less MPI
E. Test Windows DMP with a .BAT file from a DOS shell

Note: If a previous version of NX Nastran with HP MPI is installed on the cluster, there is no need to remove it. It is not used with Intel MPI and NX Nastran 10.

A. Set up password-less SSH with Cygwin 64-bit for each user

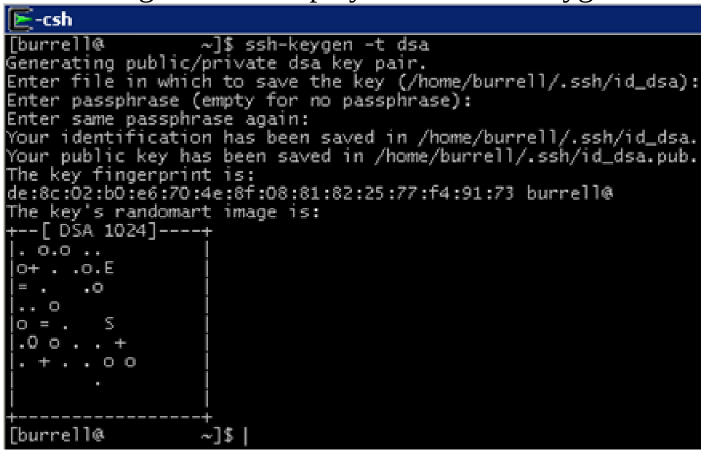
Each user must open a Cygwin64 terminal, which will place them in their home directory.

- Enter the command:

`ssh-keygen –t dsa (or, alternately with –t rsa)`

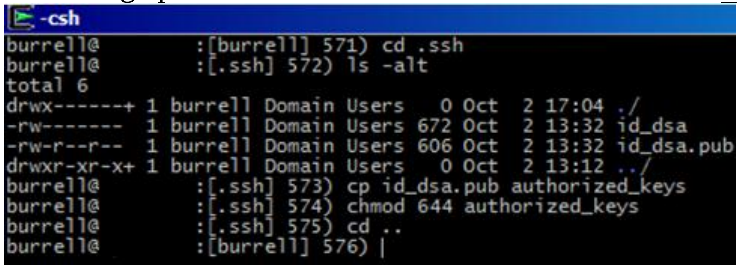
- Press Enter when you are presented a prompt.

A text image will be displayed when ssh-keygen is done:



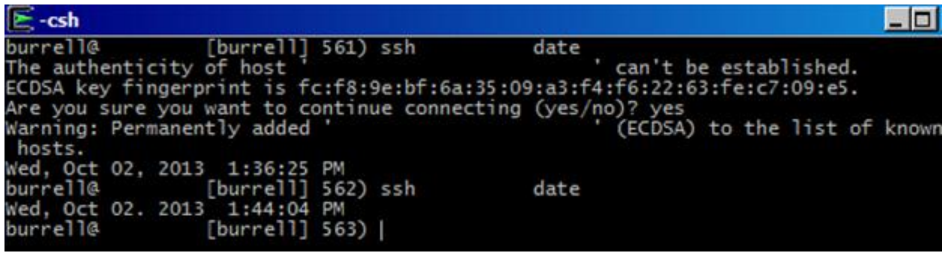
```
[burrell@          ~]$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/burrell/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/burrell/.ssh/id_dsa.
Your public key has been saved in /home/burrell/.ssh/id_dsa.pub.
The key fingerprint is:
de:8c:02:b0:e6:70:4e:8f:08:81:82:25:77:f4:91:73 burrell@
The key's randomart image is:
+--[ DSA 1024]----+
|. o.o ..         |
|o+ . .o.E        |
|= .    .o        |
|.. o             |
|o = .    S       |
|.O o . . +       |
|. + . . o o      |
|        .        |
|                 |
+-----------------+
[burrell@          ~]$ |
```

- Enter the .ssh directory with: `cd .ssh`
- Display .ssh directory contents with: `ls –alt`
- Copy the public key with: `cp id_dsa.pub authorized_keys`
- Change permissions with: `chmod 644 authorized_keys`



```
burrell@         :[burrell] 571) cd .ssh
burrell@         :[.ssh] 572) ls -alt
total 6
drwx------+ 1 burrell Domain Users   0 Oct  2 17:04 ./
-rw-------  1 burrell Domain Users 672 Oct  2 13:32 id_dsa
-rw-r--r--  1 burrell Domain Users 606 Oct  2 13:32 id_dsa.pub
drwxr-xr-x+ 1 burrell Domain Users   0 Oct  2 13:12 ../
burrell@         :[.ssh] 573) cp id_dsa.pub authorized_keys
burrell@         :[.ssh] 574) chmod 644 authorized_keys
burrell@         :[.ssh] 575) cd ..
burrell@         :[burrell] 576) |
```

- The first time you run 'ssh hostname date', answer 'yes' if prompted. All subsequent ssh commands to hostname will be password-less, as shown below.



```
burrell@[redacted] [burrell] 561) ssh [redacted] date
The authenticity of host '[redacted]' can't be established.
ECDSA key fingerprint is fc:f8:9e:bf:6a:35:09:a3:f4:f6:22:63:fe:c7:09:e5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[redacted]' (ECDSA) to the list of known
 hosts.
Wed, Oct 02, 2013  1:36:25 PM
burrell@[redacted] [burrell] 562) ssh [redacted] date
Wed, Oct 02. 2013  1:44:04 PM
burrell@[redacted] [burrell] 563) |
```

- The final step in this process is to provide password-less ssh between all nodes for each user. In order to provide password-less ssh between any two nodes, each user needs to collect all of the public keys created in the `/home/username/.ssh` directories and insert them into `/home/username/.ssh/authorized_keys` on all the nodes.

  For example, if user harry created the following public keys on these nodes:
  Master: `C:\cygwin64\home\harry\.ssh\id_dsa.pub`
  Slave1: `C:\cygwin64\home\harry\.ssh\id_dsa.pub`
  Slave2: `C:\cygwin64\home\harry\.ssh\id_dsa.pub`

  Then, user harry must insert the contents of those 3 public keys into authorized_keys on the following nodes:
  Master: `C:\cygwin64\home\harry\.ssh\authorized_keys`
  Slave1: `C:\cygwin64\home\harry\.ssh\authorized_keys`
  Slave2: `C:\cygwin64\home\harry\.ssh\authorized_keys`

  When all public keys for all the nodes are contained in the authorized_keys file on each node, then any node may run ssh between any other node without being prompted for a password.

  Note that the permissions for authorized_keys must be set to 0600 from a Cygwin64 Terminal.

  Also note that if a user's home directory is shared to all of the nodes, then it is only necessary to run ssh-keygen once, and create a single authorized_keys file in the user's `$HOME/.ssh/ directory`.
  Even though a node-name will appear at the end of the security string in the public key file (e.g., id_dsa.pub), which will be the name of the node on which 'ssh-keygen' was launched, it is merely a comment and is ignored by SSH.
  If the home directories are not shared among the nodes, then it is necessary to run ssh-keygen on each node and copy all of the public keys generated into the authorized_keys file on each node.

B. Install NX Nastran 10 to each Windows node

- Install NX Nastran 10 on all the nodes of the cluster in the same location on each node.

- Using your Webkey account, download the NX Nastran installation (.zip file) from https://download.industrysoftware.automation.siemens.com/.

- Unzip the installation file and launch `autorun`. Install to any directory that does not have spaces in the path name, for example, `D:\NXNr`. Do not install to the default path of `C:\Program Files(x86)`. Spaces in the path will prevent several Environment variables from being properly interpreted.

C. Set up Intel MPI Windows Service on each node

- Login as local Administrator and open a Command Prompt.

- Change directory with:
  ```
  cd /d D:\NXN10\nxn10\em64tnt\impi\bin
  ```

- Type the following command:
  ```
  hydra_service.exe -install
  ```

- Verify that the service is running under Administrative Tools/Services

D. Set up password-less MPI for each user on each node

Each user must enter the following commands at a Command Prompt on each node of the cluster:

```
set MPI_ROOT=D:\NXN10\nxn10\em64tnt\impi

set PATH=D:\NXN10\nxn10\em64tnt\impi\bin;%PATH%

cd /d D:\NXN10\nxn10\em64tnt\mpi\bin

mpiexec.hydra.exe -register
```

E. Test Windows DMP with a .BAT script

- Use a text editor to create the following .BAT file, replacing `node`*`i`* with the actual host names available on your cluster. Name the file `dmptest.BAT`.

```
@ECHO OFF
set NXN_BASE=D:\NXN10
%NXN_BASE%\bin\nastran.exe
"//node0/Workdir/UserName/plan10g.dat"
out="//node0/Workdir/UserName"
mem=130mw dmp=2 hosts=node0;node1
sdir=D:\Scratch slaveout=yes scr=yes
```

- Open a DOS shell, then `cd \\node0\Workdir\UserName`, where `node0`, `Workdir` and `UserName` are replaced with correct values for your hosts.

- Copy a test file from the NX Nastran tpl to your current directory

```
copy D:\NXN10\nxn10\nast\tpl\plan10g.dat
```

- Execute the .BAT file:

```
dmptest.bat
```

7.5 Determining Hosts

The nastran command uses the following hierarchy to determine the list of hosts to use:

- The nastran command "hosts" keyword on the command line
- The nastran command "hosts" keyword in an RC file.
- The local host.

Consider the following example:
The following job will run on the local host:

```
nxnr nastran example dmparallel=4
```

The following job will run on the first four available nodes from the set "node1", "node2", "node3", "node4", "node5".

```
nxnr nastran example dmparallel=4 hosts=node1:node2:node3:node4:node5
```

The following job reads the file "my.host.list".

```
nxnr nastran example dmparallel=4 hosts=my.host.list
```

The nastran command provides a simple host allocation method. If a host listed by the "hosts" keyword is unavailable, it will be skipped and the next host considered. As long as at least the number of processors specified by the "dmparallel" keyword are available on one or more of the listed hosts, the job will be allowed to run.

Hosts on Linux

On Linux systems, the "hosts" keyword needs to specify the host name of the compute nodes. A Linux example of job submittal is:

```
nxnr nastran example dmparallel=4 hosts=n1:n2:n3:n4
```

Using the "hosts" Keyword (Distributed Jobs Under LSF)

The "hosts" keyword will default to the value set by LSF when running as a distributed job and no other value for "hosts" was set on the command line or in an RC file.
Example:

```
bsub -n 4 nxnr nastran example dmp=4
```

This job will use four hosts selected by LSF. Note, the number of tasks appears twice: once for use by LSF, and once for use by NX Nastran.

Using PBS with NX Nastran

Portable Batch System (PBS) is a queuing system that can be used to submit NX Nastran serial and DMP jobs. Once you have downloaded and installed PBS, you can use the following sample script to run an NX Nastran DMP job under PBS.

The “hosts” keyword will default to the value set by PBS_NODEFILE when running as a distributed job and no other value for “hosts” was set on the command line or in an RC file.

```
#!/bin/ksh
#
# pbs_nast: PBS script to use with NX Nastran
#
# Usage: qsub -lnodes=Number-Of-Nodes pbs-nast
#
# Assume the data file is located in the directory whence the qsub
# command was issued.
#
dat=$PBS_O_WORKDIR/d10101d.dat
#
jobdat=${dat##*/}
#
# Change the working directory to the scratch directory.
#
TMPDIR=/scratch
cd $TMPDIR
#
# Pull the bulk data file over.
#
rcp $PBS_O_HOST:$dat .
#
# Determine the number of ranks.
#
dmparallel=$(sed -n -e ‘$=’ $PBS_NODEFILE)
#
# Build the hosts keyword value.
#
hostskwd=’sed -e :a -e ‘/$/N; s/\n/:/; ta’ $PBS_NODEFILE’
#
# Run the NX Nastran job. If using version 4.1 or above, comment the
following
# two lines and uncomment the next two commented lines.
#
nxnr nastran $jobdat dmparallel=$dmparallel hosts=$hostskwd \
scratch=yes batch=no
#
# If using version 4.1 or above, uncomment the following two commented
lines
```

```
# comment out the two lines preceeding the comment lines here
#
# nxnr nastran $jobdat dmparallel=$dmparallel hosts=$PBS_NODEFILE \
# scratch=yes batch=no
#
#
#
# Push the files back to the submitting host.
#
jobout=${jobdat%.*}
out=${dat%/*}
rcp -p $jobout.log $PBS_O_HOST:$out
rcp -p $jobout.f04 $PBS_O_HOST:$out
rcp -p $jobout.f06 $PBS_O_HOST:$out
rcp -p $jobout.op2 $PBS_O_HOST:$out
#
# END
```

Note: Be aware that in order to receive your job's stdout and stderr, your .rhosts file on the node issuing the "qsub" command must permit access from the remote host(s).

7.6 Managing Host-Database Directory Assignments

The performance of the disk subsystem containing the permanent end SCRATCH DBSets can have a significant impact on NX Nastran performance. In the case of a DMP job, the impact can be even greater if multiple tasks are using the same file system. To allow unique directories to be assigned to each task, the "dbs", "hosts", and "sdirectory" keywords are treated as lists scanned in a round-robin order. With this feature, you can finely control the use of disk I/O access paths by your job.

The following examples show the effect of the round-robin ordering.
nxnr nastran example dmparallel=4 hosts=a:b
sdirectory=/aa:/ba:/ab:/bb dbs=/aa:/ba:/ab:/bb

This example will assign the following host-sdirectory pairs (assuming hosts "a" and "b" each have at least two processors):

Task	Host	Scratch Directory	DBS Directory
1	a	/aa	/aa
2	b	/ba	/ba
3	a	/ab	/ab
4	b	/bb	/bb

If directory "/ba" was not available for writing by you on host "b", the tasks assignments would be (assuming host "a" has at least three processors):

Task	Host	Scratch Directory	DBS Directory
1	a	/aa	/aa
2	a	/ab	/ab
3	b	/bb	/bb
4	a	/aa	/aa

7.7 Managing Files

When an NX Nastran DMP job is running, the input file is directly read by each MPI task that can read the file, for example, via NFS. Each host that cannot read the input file will read a local copy of the file that is copied, via rcp(1), to the job's scratch directory ("sdirectory" keyword) before the job begins.

A similar check is made for the output directory. Any host that can write to the output directory ("out" keyword) will directly write its .f04, .f06, .log and other default output files to that directory. Any host that cannot see the output directory will write its default output files to the job's scratch directory. These files will then be copied, again via rcp(1), back to the output directory at the end of the job.

Note: The nastran command will perform these tests by converting your pathname value to an absolute pathname. As a result, a path that varies depending upon the host will be labeled as unreadable.

If the "sdirectory" keyword is not specified on the command line or in an RC file on the local host, each master or slave host will use its own scratch directory. This directory is determined on the master and each slave host by examining its command initialization file and version-specific RC files if the "version" keyword was defined.
Do not use an ASSIGN statement for any file that will be written by NX Nastran in a Distributed Memory Parallel (DMP) job. Instead, use the "sdirectory" and "dbs" keywords to specify names of the SCRATCH and permanent DB Sets.

7.8 Performance Issues

In addition to the normal performance issues associated with a serial or SMP job, a DMP job adds communication bandwidth as a critical performance characteristic. The basic communications channels are:

- Shared memory - SMP and NUMA systems.
- Interconnect, adapter, or switch - NUMA and distributed systems.
- High-speed special-purpose network, for example, HIPPI - all systems.
- TCP/IP network - all systems.
- Infiniband, Myrinet and Quadrics network connections on Linux are supported out of the box.

The performance of any NX Nastran job depends upon CPU, memory subsystem, and I/O subsystem performance. A Distributed Memory Parallel (DMP) job on an SMP or NUMA system is extremely sensitive to I/O subsystem performance since each task independently accesses the I/O subsystem.
If you select independent disks (not in RAID configuration) for your scratch drives, you are encouraged on SMP and NUMA systems to partition your scratch directory and

database assignments on DMP jobs using the “sdirectory” and “dbs” nastran command keywords.

Example 1:

```
nxnr nastran example dmp=4
sdir=/scr1:/scr2:/scr3:/scr4\ dbs=/dbs1:/dbs2:/dbs3:/dbs4
```

The following assignments will be made in this job:

Task	sdirectory	dbs
1	/scr1	/dbs1
2	/scr2	/dbs2
3	/scr3	/dbs3
4	/scr4	/dbs4

The preceding example will perform substantially better than the following job, which uses the default assignments for the “sdirectory” and the “dbs” keywords.

Example 2:

```
install-dir\nxnr nastran example dmp=4
```

While the ultimate effect of the communications channel on job performance is dependent upon the solution sequence, for best overall job performance, you should try to use the fastest communications channels available.

7.9 Overview of Running a DMP job

This section gives a brief overview of running DMP jobs. See the other sections of this guide for more complete details on this topic.

Command-line Syntax

You can start an NX Nastran DMP job using the command:

```
$ nxnr nastran example dmp=2 hosts=n1:n2
```

where “n1:n2” indicates the hosts to be used in the run.
Valid output is:

```
Determining available hosts, please wait...
DMP task 1: host="ugsclust1" sdir="/scratch" dbs="/scratch/plan10g";
DMP task 2: host="node1.local"; sdir="/scratch";
dbs="/scratch/plan10g";
NX Nastran beginning distributed job plan10g.
NX Nastran V4.0 (Intel Linux 2.4.20-20.7smp) Mon May 23 11:21:37 2005
NX Nastran V4.0 (Intel Linux 2.4.20-20.7bigmem) Mon May 23 11:21:37
2005
NX Nastran beginning child job plan10g.T22389_37.t1 on node1.local.
NX Nastran beginning child job plan10g.t0 on ugsclust1 (master).
```

Note: The “beginning child job” lines may appear in a random order.

Error Examples
The following error is an authorization problem.

```
*** USER FATAL MESSAGE 3060 (PREFACE)
    SUBROUTINE MODEL    - OPTION NAST NOT IN APPROVED LIST.
    SYSTEM DATE (MM/DD/YY): mm/dd/yy
    SYSTEM UGSID:  n (DECIMAL) n (HEXADECIMAL)
```

Likely causes are:

- The license or authorization file does not include the ability to make DMP runs.
- The license or authorization file was not accessible to the first node in the hosts list.

Appendix

SEQP STYLE DMP SOLUTIONS

In the parallel processing of a finite element application, finite element model data should be distributed to each processor. There are two choices of method for domain partitioning in an NX Nastran parallel processing task: grid-based partitioning and degree of freedom-based partitioning. The SEQP module performs grid-based partitioning, in which *p* domains are created by an automatic partitioner from the grid-based graph. The GPARTN module performs graph partitioning for the graph based on degrees of freedom or grids, whichever is better.

The GPARTN-based method is available in more solution sequences than the SEQP-based, and generally, the GPARTN module gives better performance than the module SEQP. For example, the GPARTN module provides better performance if the finite element model includes many Multi-Point Constraints (MPCs). It should be noted that the SEQP-based method can not handle a finite element model that includes acoustic fluid, glue elements, weld/fast elements, or modal effective mass requests. Also, grid point weight generator output reflects only the local portion of the model (rather than global) with the SEQP module, and thus the results may be unexpected.

The partitioning method can be selected by the nastran command keyword **‘gpart’** in the following way.

Module	**Keyword**	
SEQP	gpart = 0	
GPARTN	gpart = 1	default

Each module provides two different algorithms for graph partitioning. The algorithm can be selected by the DMAP parameter **oldseq**.

Module	**Method**	**DMAP parameter**	
SEQP	EXTREME	param, oldseq = 9	
	MLV	param, oldseq = 11	default
GPARTN	METIS	param, oldseq = 10	
	MLV	param, oldseq = 11	default

RDMODES, SOL111, SOL 112, and SOL 200 use GPARTN only.

GDMODES/HDMODES in SOL 103 and GDSTAT can use either partitioning module (GPARTN is the default).

The following table lists the availability of domain partitioning modules for each computational method:

	SEQP	**GPARTN**
GDSTAT	X	X(default)
GDMODES	X	X(default)
HDMODES	X	X(default)
RDMODES	Na	X(default)

In the above table, ‘X’ means the partitioning module is available. ‘na’ refers to ‘not applicable’.

Note: Neither the module SEQP or GPARTN is used in **LDSTAT**, **FDMODES,** and **FDFREQR**, because these computational methods do not require domain partitioning.

In chapter 4, we already provided examples for GDMODES and HDMODES with GPARTN. Here we present examples with SEQP.

A finite element plate model with 110 grid points and 100 CQUAD4 elements is executed in parallel with SOL 103 analysis. The total number of degrees of freedom is 660. Four processors are used for the DMP run.

1. GDMODES with SEQP
GDMODES with SEQP is implemented with the nastran keyword "gpart=0".

```
nastran example dmp=p gpart=0
```

```
nastran example dmp=4 gpart=0
```

When the domain is partitioned, the output from the SEQP module is shown as:

```
     STATISTICS FROM AUTOMATIC MODEL PARTITIONER

     THE NUMBER OF DOMAINS CREATED USING EDSMLV   IS        4
     THE NUMBER OF GRID POINTS IN THE GLOBAL BOUNDARY IS        30
     THE NUMBER OF ELASTIC ELEMENTS IN THE RESIDUAL IS         0
     THE NUMBER OF RIGID ELEMENTS IN THE RESIDUAL IS         0
 DOMAIN ID     # INTERNAL GRID POINTS     # EXTERNAL GRID POINTS     # OF ELEMENTS
 ---------     ----------------------     ----------------------     -------------
    1                   20                         16                       25
    2                   20                         16                       25
    3                   20                         16                       25
    4                   20                         16                       25

  GEOMETRY DOMAIN PARALLEL LANCZOS METHOD
```

The partitioning statistics show the number of domains created, grid points, and elements for each sub-domain.

Once the results of eigenvalue analysis on the slave processors are collected, the master processor prints the summary of merged eigensolutions.

```
EIGENVALUES FOUND IN DOMAIN # 1


      E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)


                BLOCK SIZE USED .....................    7

                NUMBER OF DECOMPOSITIONS ............    3

                NUMBER OF ROOTS FOUND ...............  208

                NUMBER OF SOLVES REQUIRED ...........   41


$--------------------------------------------------------------------------------
$   The list of all eigensolutions collected from the slave processor
$--------------------------------------------------------------------------------

                         R E A L   E I G E N V A L U E S
   MODE    EXTRACTION    EIGENVALUE     RADIANS      CYCLES    GENERALIZED  NERALIZED
    NO.       ORDER                                                MASS       STIFFNESS
     1         1          1.696349E+07   4.118676E+03   6.555076E+02  1.000000E+00   1.696349E+07
     2         2          1.848026E+07   4.298867E+03   6.841859E+02  1.000000E+00   1.848026E+07
 ⋮
     207      207         3.073819E+09   5.544203E+04   8.823873E+03  1.000000E+00   3.073819E+09
     208      208         3.123167E+09   5.588530E+04   8.894422E+03  1.000000E+00   3.123167E+09
```

With the gpart=0 option, the summary of eigenvalue analysis from the master processor is broadcasted to the slave processors, so that all processors print the same summary information.

```
 ****************
S L A V E   1
****************
 ⋮
 EIGENVALUES FOUND IN DOMAIN # 2


    E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)



              BLOCK SIZE USED .....................    7

              NUMBER OF DECOMPOSITIONS ............    3

              NUMBER OF ROOTS FOUND ...............  208

              NUMBER OF SOLVES REQUIRED ...........   41
$--------------------------------------------------------------------------------
$          The list of all eigensolutions collected from the slave processor
$--------------------------------------------------------------------------------
```

```
                           R E A L   E I G E N V A L U E S
   MODE     EXTRACTION     EIGENVALUE     RADIANS       CYCLES     GENERALIZED   NERALIZED
    NO.        ORDER                                                   MASS        STIFFNESS
     1           1          1.696349E+07      4.118676E+03   6.555076E+02  1.000000E+00    1.696349E+07
     2           2          1.848026E+07      4.298867E+03   6.841859E+02  1.000000E+00    1.848026E+07
 .
 .
 .
     207       207          3.073819E+09       5.544203E+04  8.823873E+03  1.000000E+00    3.073819E+09
     208       208          3.123167E+09       5.588530E+04  8.894422E+03  1.000000E+00    3.123167E+09
```

2.HDMODES with SEQP

HDMODES with SEQP is implemented with the nastran keyword "gpart=0".

```
nastran example dmp=p nclust=c gpart=0
```

In the example below, HDMODES is executed with ‘dmp=4’ and ‘nclust=2’ keywords. Among the four processors, the first processor is the master processor, and the other processors are the slave processors.

Note that HDMODES defines the local master and local slave processors inside of each cluster.

nastran example dmp=4 nclust =2 gpart=0

The partitioning statistics show the information of domains for each cluster. There are two domains for each cluster. The first domain of the cluster has 50 internal grids and 10 external grids, and the second domain has the same local size. Generally, the size of each domain is different.

```
      STATISTICS FROM AUTOMATIC MODEL PARTITIONER

      THE NUMBER OF DOMAINS CREATED USING EDSMLV   IS        2
      THE NUMBER OF GRID POINTS IN THE GLOBAL BOUNDARY IS         10
      THE NUMBER OF ELASTIC ELEMENTS IN THE RESIDUAL IS          0
      THE NUMBER OF RIGID ELEMENTS IN THE RESIDUAL IS          0
      DOMAIN ID     # INTERNAL GRID POINTS     # EXTERNAL GRID POINTS     # OF ELEMENTS
      ---------     ----------------------     ----------------------     -------------
          1                   50                         10                      50
          2                   50                         10                      50


   HIERARCHIC DOMAIN PARALLEL LANCZOS METHOD
```

In this example, cluster 1 includes processors 1 and 3, and processors 2 and 4 are in cluster 2. Processors 1 and 2 are the local masters in clusters 1 and 2, respectively. In cluster 1, 155 eigenvalues are collected in processor 1. Similarly, in cluster 2, 53 eigenvalues are collected in processor 2. Once each local master processor collects the results for each cluster, the master processor (processor 1) collects the information from the local masters.

In the .f06 file, the master processor prints the summary of the eigenvalue analysis and the list of eigenvalues. Be careful when interpreting the "number of roots found" information in the summary. This information concerns the cluster in which the master is included. For example, in the model below, interpret the "number of roots found" message from the eigenvalue analysis summary as indicating that cluster 1 found 155 eigenvalues. The master processor eigenvalue table lists all 208 eigenvalues that are merged from all of the local master processors. You should determine the number of modes from the list of eigenvalues.

```
EIGENVALUES FOUND IN DOMAIN # 1


           E I G E N V A L U E   A N A L Y S I S   S U M M A R Y   (READ MODULE)



                     BLOCK SIZE USED ......................    7

                     NUMBER OF DECOMPOSITIONS ............    3

                     NUMBER OF ROOTS FOUND ...............  155

                     NUMBER OF SOLVES REQUIRED ...........   41

$--------------------------------------------------------------------------------
$          The list of all eigensolutions collected from the slave processor
$--------------------------------------------------------------------------------


                           R E A L   E I G E N V A L U E S
   MODE    EXTRACTION    EIGENVALUE    RADIANS     CYCLES   GENERALIZED  NERALIZED
    NO.       ORDER                                            MASS       STIFFNESS
     1          1      1.696349E+07   4.118676E+03  6.555076E+02  1.000000E+00  1.696349E+07
     2          2      1.848026E+07   4.298867E+03  6.841859E+02  1.000000E+00  1.848026E+07
 ⋮

    207        207      3.073819E+09   5.544203E+04  8.823873E+03  1.000000E+00  3.073819E+09
    208        208      3.123167E+09   5.588530E+04  8.894422E+03  1.000000E+00  3.123167E+09
```

Again, be cautious when interpreting the output from slave processors. The summary of eigenvalue analysis for slave processors is confined to the corresponding processor. In the list of eigenvalues on the slave processors, the mode number does not represent the global mode number.

Note that, with the gpart=0, the master process broadcasts the collected output only to the local masters. In the example below, the local slave, processor 2, prints 208 eigenvalues, while another local slave, processor 4, prints 53 eigenvalues.

```
 ⋮
****************
  S L A V E   1
  ****************
  ⋮
53  EIGENVALUES FOUND IN DOMAIN # 1


        E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)



                  BLOCK SIZE USED .....................    7

                  NUMBER OF DECOMPOSITIONS ............    2

                  NUMBER OF ROOTS FOUND ...............    53

                  NUMBER OF SOLVES REQUIRED ...........   39




                        R E A L   E I G E N V A L U E S
    MODE    EXTRACTION    EIGENVALUE    RADIANS     CYCLES   GENERALIZED  NERALIZED
     NO.       ORDER                                             MASS        STIFFNESS
1        1        1.632620E+09   4.040570E+04  6.430766E+03  1.000000E+00  1.632620E+09
2        2        1.633399E+09   4.041533E+04  6.432300E+03  1.000000E+00  1.633399E+09
  ⋮


    52       52       3.073819E+09   5.544203E+04  8.823873E+03  1.000000E+00  3.073819E+09
    53       53       3.123167E+09   5.588530E+04  8.894422E+03  1.000000E+00  3.123167E+09


****************
  S L A V E   2
****************
  ⋮
155  EIGENVALUES FOUND IN DOMAIN # 2


          E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)



                    BLOCK SIZE USED .....................    7

                    NUMBER OF DECOMPOSITIONS ............    3

                    NUMBER OF ROOTS FOUND ...............  155

                    NUMBER OF SOLVES REQUIRED ...........   41


                        R E A L   E I G E N V A L U E S
    MODE    EXTRACTION    EIGENVALUE    RADIANS     CYCLES   GENERALIZED  NERALIZED
    NO.       ORDER                                             MASS        STIFFNESS
    1        1       1.696349E+07    4.118676E+03  6.555076E+02  1.000000E+00  1.696349E+07
    2        2        1.848026E+07   4.298867E+03   6.841859E+02  1.000000E+00 1.848026E+07
  ⋮
    207      207      3.073819E+09   5.544203E+04   8.823873E+03  1.000000E+00 3.073819E+09
    208      208      3.123167E+09   5.588530E+04   8.894422E+03  1.000000E+00 3.123167E+09


****************
  S L A V E   3
 ****************
 ⋮
53  EIGENVALUES FOUND IN DOMAIN # 2

         E I G E N V A L U E  A N A L Y S I S   S U M M A R Y   (READ MODULE)

                   BLOCK SIZE USED .....................    7

                   NUMBER OF DECOMPOSITIONS ............    2

                   NUMBER OF ROOTS FOUND ...............   53

                   NUMBER OF SOLVES REQUIRED ...........   39
```

```
                     R E A L   E I G E N V A L U E S
   MODE    EXTRACTION    EIGENVALUE     RADIANS      CYCLES   GENERALIZED  NERALIZED
    NO.       ORDER                                              MASS       STIFFNESS
    1          1       1.632620E+09     4.040570E+04  6.430766E+03 1.000000E+00  1.632620E+09
    2          2       1.633399E+0      4.041533E+04  6.432300E+03 1.000000E+00  1.633399E+09
⋮

    52         52      3.073819E+09      5.544203E+04  8.823873E+03 1.000000E+00  3.073819E+09
      53      3.123167E+09      5.588530E+04  8.894422E+03 1.000000E+00  3.123167E+09
```

3. GDSTAT with SEQP

GDSTAT with SEQP is implemented with the nastran keyword “gpart=0”.

```
nastran example dmp=p gpart=0
```

Example:

`nastran example dmp=4 gpart=0`

The static analysis is performed for a finite element cube model that has 2197 grid points for 1728 CHEXA elements and 6048 CQUAD4 elements. GDSTAT is implemented in a DMP run. The module SEQP partitioned the global finite element model into four sub-domains and a boundary. The following information in the .f06 describes the detailed statistics of partitioning.

```
        STATISTICS FROM AUTOMATIC MODEL PARTITIONER

     THE NUMBER OF DOMAINS CREATED USING EXTREME  IS             4
     THE NUMBER OF GRID POINTS IN THE GLOBAL BOUNDARY IS       361
     THE NUMBER OF ELASTIC ELEMENTS IN THE RESIDUAL IS           0
     THE NUMBER OF RIGID ELEMENTS IN THE RESIDUAL IS             0
     DOMAIN ID     # INTERNAL GRID POINTS     # EXTERNAL GRID POINTS     # OF ELEMENTS
     ---------     ----------------------     ----------------------     -------------
           1                  468                        174                    2060
           2                  468                        169                    2016
           3                  468                        169                    1880
           4                  432                        205                    1820
```

Each processor performs static analysis with the corresponding local domain and the boundary, and the master processor collects the results that you requested through communications. These outputs are printed only on the master processor.

```
                                          D I S P L A C E M E N T   V E C T O R

POINT ID.   TYPE          T1             T2             T3             R1             R2             R3
   30060      G      1.578281E-05  -9.678060E-05  -2.301274E-04  -4.363790E-06  -3.554464E-06  -4.259573E-
09
   40066      G     -1.578241E-05  -9.678057E-05  -2.301267E-04  -4.363756E-06   3.554413E-06   4.257577E-
09
   70660      G     -1.578268E-05   9.677987E-05  -2.301273E-04  -4.363777E-06  -3.554452E-06   4.259614E-
09
   80666      G      1.578229E-05   9.677984E-05  -2.301266E-04  -4.363744E-06   3.554401E-06  -4.257619E-
09
```

It is strongly recommended to use the nastran keyword "slaveout=yes" to print out analysis procedures from all processors. With "slaveout=yes", the f04, .f06, and .log files contain the outputs of master and slave processors in the following format.

```
⋮
 }master processor
                              * * * END OF JOB * * *

****************
  S L A V E   1
 ****************
                                             ⋮
 }  slave 1 processor
                              * * * END OF JOB * * *

 ****************
  S L A V E   2
 ****************
                                             ⋮
 }  slave 2 processor
                              * * * END OF JOB * * *

 ****************
  S L A V E   3
 ****************
                                             ⋮
 }  slave 3 processor
                              * * * END OF JOB * * *
```

References

[1] L. Komzsik, The Lanczos Method: Evolution and Application, 2003, SIAM.

[2] W. Gropp, E. Lusk, and A. Skjellum, Using MPI: Portable Parallel Programming with the Message Passing Interface, 1997, MIT.

[3] NX Nastran Numerical Methods User’s Guide.