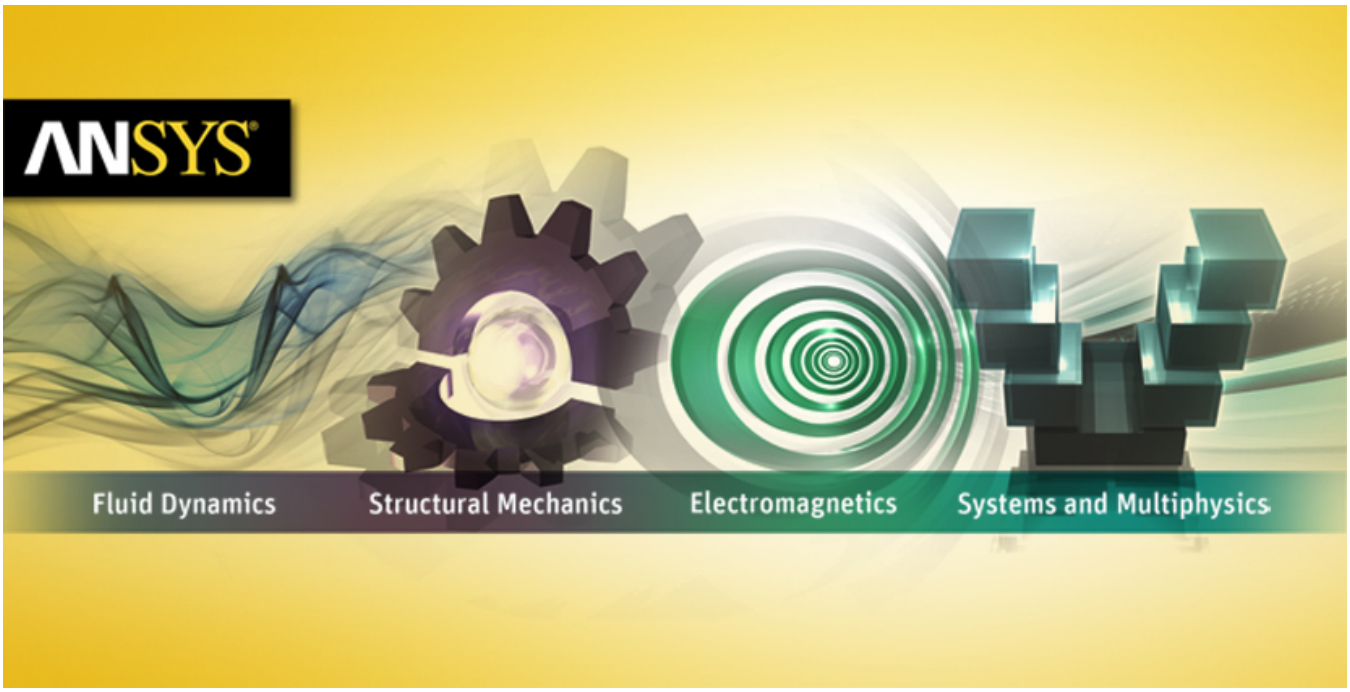

ANSYS Mechanical APDL Parallel Processing Guide

ANSYS, Inc.
Southpointe
275 Technology Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
http://www.ansys.com
(T) 724-746-3304
(F) 724-514-9494

Release 15.0
November 2013

ANSYS, Inc. is certified to ISO 9001:2008.

Copyright and Trademark Information

Disclaimer Notice

U.S. Government Rights

Third-Party Software

Published in the U.S.A.

Table of Contents

List of Tables

Chapter 1: Overview of Parallel Processing

Solving a large model with millions of DOFs or a medium-sized model with nonlinearities that needs many iterations to reach convergence can require many CPU hours. To decrease simulation time, ANSYS, Inc. offers different parallel processing options that increase the model-solving power of ANSYS products by using multiple processors (also known as cores). The following three parallel processing capabilities are available:

- Shared-memory parallel processing (shared-memory ANSYS)
- Distributed-memory parallel processing (Distributed ANSYS)
- GPU acceleration (a type of shared-memory parallel processing)

Multicore processors, and thus the ability to use parallel processing, are now widely available on all computer systems, from laptops to high-end servers. The benefits of parallel processing are compelling but are also among the most misunderstood. This chapter explains the two types of parallel processing available in ANSYS and also discusses the use of GPUs (considered a form of shared-memory parallel processing) and how they can further accelerate the time to solution.

Currently, the default scheme is to use up to two cores with shared-memory parallelism. For many of the computations involved in a simulation, the speedups obtained from parallel processing are nearly linear as the number of cores is increased, making very effective use of parallel processing. However, the total benefit (measured by elapsed time) is problem dependent and is influenced by many different factors.

No matter what form of parallel processing is used, the maximum benefit attained will always be limited by the amount of work in the code that cannot be parallelized. If just 20 percent of the runtime is spent in nonparallel code, the maximum theoretical speedup is only 5X, assuming the time spent in parallel code is reduced to zero. However, parallel processing is still an essential component of any HPC system; by reducing wall clock elapsed time, it provides significant value when performing simulations.

Both Distributed ANSYS and shared-memory ANSYS can require HPC licenses. Distributed ANSYS and shared-memory ANSYS allow you to use two cores without using any HPC licenses. Additional licenses will be needed to run with more than two cores. The GPU accelerator capability always requires an HPC license. Several HPC license options are available. See HPC Licensing (p. 3) for more information.

ANSYS LS-DYNA If you are running ANSYS LS-DYNA, you can use LS-DYNA's parallel processing (MPP or SMP) capabilities. Use the launcher method or command line method as described in Activating Distributed ANSYS (p. 27) to run LS-DYNA MPP. Also see LS-DYNA Parallel Processing Capabilities in the *ANSYS LS-DYNA User's Guide* for more information on both the SMP and MPP capabilities. You will need an ANSYS LS-DYNA Parallel license for every core beyond the first one.

1.1. Parallel Processing Terminolgy

It is important to fully understand the terms we use, both relating to our software and to the physical hardware. The terms *shared-memory ANSYS* and *Distributed ANSYS* refer to our software offerings, which run on shared-memory or distributed-memory hardware configurations. The term *GPU accelerator cap-*

ability refers to our software offering which allows the program to take advantage of certain GPU (graphics processing unit) hardware to accelerate the speed of the solver computations.

1.1.1. Hardware Terminology

The following terms describe the hardware configurations used for parallel processing:

Shared-memory hardware	This term refers to a physical hardware configuration in which a single shared-memory address space is accessible by multiple CPU cores; each CPU core "shares" the memory with the other cores. A common example of a shared-memory system is a Windows desktop machine or workstation with one or two multicore processors.
Distributed-memory hardware	This term refers to a physical hardware configuration in which multiple machines are connected together on a network (i.e., a cluster). Each machine on the network (that is, each compute node on the cluster) has its own memory address space. Communication between machines is handled by interconnects (Gigabit Ethernet, Myrinet, Infiniband, etc.). Virtually all clusters involve both shared-memory and distributed-memory hardware. Each compute node on the cluster typically contains at least two or more CPU cores, which means there is a shared-memory environment within a compute node. The distributed-memory environment requires communication between the compute nodes involved in the cluster.
GPU hardware	A graphics processing unit (GPU) is a specialized microprocessor that offloads and accelerates graphics rendering from the microprocessor. Their highly parallel structure makes GPUs more effective than general-purpose CPUs for a range of complex algorithms. In a personal computer, a GPU on a dedicated video card is more powerful than a GPU that is integrated on the motherboard.

1.1.2. Software Terminology

The following terms describe our software offerings for parallel processing:

Shared-memory ANSYS	This term refers to running across multiple cores on a single machine (e.g., a desktop workstation or a single compute node of a cluster). Shared-memory parallelism is invoked, which allows each core involved to share data (or memory) as needed to perform the necessary parallel computations. When run within a shared-memory architecture, most computations in the solution phase and many pre- and postprocessing operations are performed in parallel. For more information, see Using Shared-Memory ANSYS (p. 5).
Distributed ANSYS	This term refers to running across multiple cores on a single machine (e.g., a desktop workstation or a single compute node of a cluster) or across multiple machines (e.g., a cluster). Distributed-memory parallelism is invoked, and each core communicates data needed to perform the necessary parallel computations through the use of MPI (Message Passing Interface) software. With Distributed ANSYS, all computations in the solution phase are performed in parallel (including the stiffness matrix

generation, linear equation solving, and results calculations). Pre- and postprocessing do not make use of the distributed-memory parallel processing; however, these steps can make use of shared-memory parallelism. See Using Distributed ANSYS (p. 17) for more details.

GPU accelerator capability — This capability takes advantage of the highly parallel architecture of the GPU hardware to accelerate the speed of solver computations and, therefore, reduce the time required to complete a simulation. Some computations of certain equation solvers can be off-loaded from the CPU(s) to the GPU, where they are often executed much faster. The CPU core(s) will continue to be used for all other computations in and around the equation solvers. For more information, see GPU Accelerator Capability (p. 9).

Shared-memory ANSYS can only be run on shared-memory hardware. However, Distributed ANSYS can be run on both shared-memory hardware or distributed-memory hardware. While both forms of hardware can achieve a significant speedup with Distributed ANSYS, only running on distributed-memory hardware allows you to take advantage of increased resources (for example, available memory and disk space, as well as memory and I/O bandwidths) by using multiple machines.

Currently, only a single GPU accelerator device per machine (e.g., desktop workstation or single compute node of a cluster) can be utilized during a solution. The GPU accelerator capability can be used with either shared-memory ANSYS or Distributed ANSYS.

1.2. HPC Licensing

ANSYS, Inc. offers the following high performance computing license options:

ANSYS HPC - These physics-neutral licenses can be used to run a single analysis across multiple processors (cores).
ANSYS HPC Packs - These physics-neutral licenses share the same characteristics of the ANSYS HPC licenses, but are combined into predefined packs to give you greater value and scalability.
Physics-Specific Licenses - Legacy physics-specific licenses are available for various applications. The physics-specific license for Distributed ANSYS and shared-memory ANSYS is the ANSYS Mechanical HPC license.

For detailed information on these HPC license options, see HPC Licensing in the *ANSYS, Inc. Licensing Guide*.

The HPC license options cannot be combined with each other in a single solution; for example, you cannot use both ANSYS HPC and ANSYS HPC Packs in the same analysis solution.

The order in which HPC licenses are used is specified by your user license preferences setting. See Specifying HPC License Order in the *ANSYS, Inc. Licensing Guide* for more information on setting user license preferences.

Both Distributed ANSYS and shared-memory ANSYS allow you to use two non-GPU cores without using any HPC licenses. ANSYS HPC licenses and ANSYS Mechanical HPC licenses add cores to this base functionality, while the ANSYS HPC Pack licenses function independently of the two included cores.

GPU acceleration is allowed when using ANSYS HPC physics neutral licenses or ANSYS HPC Pack licenses with Mechanical APDL or with the Mechanical Application. The combined number of CPU and GPU processors used cannot exceed the task limit allowed by your specific license configuration.

The HPC license options described here do not apply to ANSYS LS-DYNA; see the *ANSYS LS-DYNA User's Guide* for details on parallel processing options with ANSYS LS-DYNA.

Chapter 2: Using Shared-Memory ANSYS

When running a simulation, the solution time is typically dominated by three main parts: the time spent to create the element matrices and form the global matrices, the time to solve the linear system of equations, and the time spent calculating derived quantities (such as stress and strain) and other requested results for each element.

Shared-memory ANSYS can run a solution over multiple cores on a single machine. When using shared-memory parallel processing, you can reduce each of the three main parts of the overall solution time by using multiple cores. However, this approach is often limited by the memory bandwidth; you typically see very little reduction in solution time beyond four cores.

The main program functions that run in parallel on shared-memory hardware are:

- Solvers such as the Sparse, PCG, ICCG, Block Lanczos, PCG Lanczos, Supernode, and Subspace running over multiple processors but sharing the same memory address. These solvers typically have limited scalability when used with shared-memory parallelism. In general, very little reduction in time occurs when using more than four cores.
- Forming element matrices and load vectors.
- Computing derived quantities and other requested results for each element.
- Pre- and postprocessing functions such as graphics, selecting, sorting, and other data and compute intensive operations.

2.1. Activating Parallel Processing in a Shared-Memory Architecture

1. Shared-memory ANSYS uses two cores by default and does not require any HPC licenses. Additional HPC licenses are required to run with more than two cores. Several HPC license options are available. See HPC Licensing for more information.
2. Open the Mechanical APDL Product Launcher:

   Windows:
   Start >Programs >ANSYS 15.0 >Mechanical APDL Product Launcher

   Linux:

   ```
   launcher150
   ```

3. Select the correct environment and license.
4. Go to the **High Performance Computing Setup** tab. Select **Use Shared-Memory Parallel (SMP)**. Specify the number of cores to use.
5. Alternatively, you can specify the number of cores to use via the `-np` command line option:

   ```
   ansys150 -np N
   ```

where *N* represents the number of cores to use.

For large multiprocessor servers, ANSYS, Inc. recommends setting *N* to a value no higher than the number of available cores *minus one*. For example, on an eight-core system, set *N* to 7. However, on multiprocessor workstations, you may want to use all available cores to minimize the total solution time. The program automatically limits the maximum number of cores used to be less than or equal to the number of physical cores on the machine. This is done to avoid running the program on virtual cores (e.g., by means of hyperthreading), which typically results in poor per-core performance. For optimal performance, consider closing down all other applications before launching ANSYS.

6. If working from the launcher, click **Run** to launch ANSYS.

7. Set up and run your analysis as you normally would.

2.1.1. System-Specific Considerations

For shared-memory parallel processing, the number of cores that the program uses is limited to the *lesser* of one of the following:

- The number of ANSYS Mechanical HPC licenses available (plus the first two cores which do not require any licenses)
- The number of cores indicated via the `-np` command line argument
- The actual number of cores available

You can specify multiple settings for the number of cores to use during a session. However, ANSYS, Inc. recommends that you issue the **/CLEAR** command before resetting the number of cores for subsequent analyses.

2.2. Troubleshooting

This section describes problems which you may encounter while using shared-memory parallel processing as well as methods for overcoming these problems. Some of these problems are specific to a particular system, as noted.

Job fails with SIGTERM signal (Linux Only)

Occasionally, when running on Linux, a simulation may fail with the following message: "process killed (SIGTERM)". This typically occurs when computing the solution and means that the system has killed the ANSYS process. The two most common occurrences are (1) ANSYS is using too much of the hardware resources and the system has killed the ANSYS process or (2) a user has manually killed the ANSYS job (i.e., **kill -9** system command). Users should check the size of job they are running in relation to the amount of physical memory on the machine. Most often, decreasing the model size or finding a machine with more RAM will result in a successful run.

Poor Speedup or No Speedup

As more cores are utilized, the runtimes are generally expected to decrease. The biggest relative gains are typically achieved when using two cores compared to using a single core. When significant speedups are not seen as additional cores are used, the reasons may involve both hardware and software issues. These include, but are not limited to, the following situations.

Hardware

Oversubscribing hardware In a multiuser environment, this could mean that more physical cores are being used by ANSYS simulations than are available on the machine. It could also mean that hyperthreading is activated. Hyperthreading typically involves enabling extra virtual cores, which can sometimes allow software programs to more effectively use the full processing power of the CPU. However, for compute-intensive programs such as ANSYS, using these virtual cores rarely provides a significant reduction in runtime. Therefore, it is recommended you disable hyperthreading; if hyperthreading is enabled, it is recommended you do not exceed the number of physical cores.

Lack of memory bandwidth On some systems, using most or all of the available cores can result in a lack of memory bandwidth. This lack of memory bandwidth can impact the overall scalability of the ANSYS software.

Dynamic Processor Speeds Many new CPUs have the ability to dynamically adjust the clock speed at which they operate based on the current workloads. Typically, when only a single core is being used the clock speed can be significantly higher than when all of the CPU cores are being utilized. This can have a negative impact on scalability as the per-core computational performance can be much higher when only a single core is active versus the case when all of the CPU cores are active.

Software

Simulation includes non-supported features The shared- and distributed-memory parallelisms work to speed up certain compute-intensive operations in **/PREP7**, **/SOLU** and **/POST1**. However, not all operations are parallelized. If a particular operation that is not parallelized dominates the simulation time, then using additional cores will not help achieve a faster runtime.

Simulation has too few DOF (degrees of freedom) Some analyses (such as transient analyses) may require long compute times, not because the number of DOF is large, but because a large number of calculations are performed (i.e., a very large number of time steps). Generally, if the number of DOF is relatively small, parallel processing will not significantly decrease the solution time. Consequently, for small models with many time steps, parallel performance may be poor because the model size is too small to fully utilize a large number of cores.

I/O cost dominates solution time For some simulations, the amount of memory required to obtain a solution is greater than the physical memory (i.e., RAM) available on the machine. In these cases, either virtual memory (i.e., hard disk space) is used by the operating system to hold the data that would otherwise be stored in memory, or the equation solver writes extra files to the disk to store data. In both cases, the extra I/O done using the hard drive can significantly impact performance, making the I/O performance the main bottleneck to achieving optimal performance. In these cases, using additional cores will typically not result in a significant reduction in overall time to solution.

Different Results Relative to a Single Core

Shared-memory parallel processing occurs in various preprocessing, solution, and postprocessing operations. Operational randomness and numerical round-off inherent to parallelism can cause slightly different results between runs on the same machine using the same number of cores or different numbers of cores. This difference is often negligible. However, in some cases the difference is appreciable. This sort of behavior is most commonly seen on nonlinear static or transient analyses which are numerically unstable. The more numerically unstable the model is, the more likely the convergence pattern or final results will differ as the number of cores used in the simulation is changed.

With shared-memory parallelism, you can use the **PSCONTROL** command to control which operations actually use parallel behavior. For example, you could use this command to show that the element matrix generation running in parallel is causing a nonlinear job to converge to a slightly different

solution each time it runs (even on the same machine with no change to the input data). This can help isolate parallel computations which are affecting the solution while maintaining as much other parallelism as possible to continue to reduce the time to solution.

Chapter 3: GPU Accelerator Capability

In an effort to provide faster performance during solution, Mechanical APDL supports offloading key solver computations onto graphics cards to accelerate those computations. Only high-end graphics cards, the ones with the most amount of cores and memory, can be used to accelerate the solver computations. For details on which GPU devices are supported and the corresponding driver versions, see the GPU requirements outlined in the Windows Installation Guide and the Linux Installation Guide.

More recently, Intel has released the Xeon Phi series of coprocessors which are similar in design to these high-end graphics cards. For the purposes of the GPU accelerator capability, when the term "graphics card" or "GPU" is used, it can refer to an Intel Xeon Phi coprocessor as well.

It is important to understand that a GPU does not replace the CPU core(s) on which a simulation typically runs. One or more CPU cores must be used to run the Mechanical APDL program. The GPUs are used in support of the CPU to process certain calculations. The CPU continues to handle most operations and will automatically offload some of the time-intensive parallel operations performed by certain equation solvers. These parallel solver operations can usually be performed much faster on the highly parallel architecture of a GPU, thus accelerating these solvers and reducing the overall time to solution.

GPU acceleration can be used with both shared-memory parallel processing (shared-memory ANSYS) and distributed-memory parallel processing (Distributed ANSYS). In shared-memory ANSYS, one or multiple GPU accelerator devices can be utilized during solution. In Distributed ANSYS, one or multiple GPU accelerator devices *per machine or compute node* can be utilized during solution.

As an example, when using Distributed ANSYS on a cluster involving eight compute nodes with each compute node having two supported GPU accelerator devices, either a single GPU per node (a total of eight GPU cards) or two GPUs per node (a total of sixteen GPU cards) can be used to accelerate the solution. The GPU accelerator device usage must be consistent across all compute nodes. For example, if running a simulation across all compute nodes, it is not possible to use one GPU for some compute nodes and zero or two GPUs for the other compute nodes.

On machines containing multiple GPU accelerator devices, the program automatically selects the GPU accelerator device (or devices) to be used for the simulation. The program cannot detect if a GPU device is currently being used by other software, including another Mechanical APDL simulation. Therefore, in a multiuser environment, users should be careful not to oversubscribe the GPU accelerator devices by simultaneously launching multiple simulations that attempt to use the same GPU (or GPUs) to accelerate the solution. For more information, see Oversubscribing GPU Hardware in the troubleshooting discussion.

The GPU accelerator capability is only supported on the Windows 64-bit and Linux x64 platforms.

GPU acceleration is allowed when using ANSYS HPC physics neutral licenses or ANSYS HPC Pack licenses. For more information see HPC Licensing in the *ANSYS, Inc. Licensing Guide*.

The following GPU accelerator topics are available:

3.1. Activating the GPU Accelerator Capability

Following is the general procedure to use the GPU accelerator capability:

1. Before activating the GPU accelerator capability, you must have at least one supported GPU card (or Xeon Phi coprocessor) with the proper driver level and proper HPC licensing.

2. Open the Mechanical APDL Product Launcher.

   Windows:
   Start >Programs >ANSYS 15.0 >Mechanical APDL Product Launcher

   Linux:

   ```
   launcher150
   ```

3. Select the correct environment and license.

4. Go to the **High Performance Computing Setup** tab. Select **Use GPU Accelerator Capability**.

5. Alternatively, you can activate the GPU accelerator capability via the `-acc` command line option:

   ```
   ansys150 -acc nvidia -na N
   ```

   OR

   ```
   ansys150 -acc intel -na N
   ```

   The `-na` command line option followed by a number (*N*) indicates the number of GPU accelerator devices to use per machine or compute node. If only the `-acc` option is specified, the program uses a single GPU device per machine or compute node by default (that is, `-na 1`).

6. If working from the launcher, click Run to launch Mechanical APDL.

7. Set up and run your analysis as you normally would.

Note

The **High Performance Computing Setup** tab of the Product Launcher does not allow you to specify GPU acceleration in conjunction with distributed-memory parallel processing, nor does it allow you to specify multiple GPU devices per machine or compute node, nor does it allow you to enable GPU acceleration on Intel Xeon Phi coprocessors. If any of these capabilities is desired, you must select the **Customization/Preferences** tab in the Product Launcher and input the additional command line arguments (for example, `-acc intel`, `-acc nvidia`, or `-na N`) in the **Additional Parameters** field. Alternatively, you can use the command line to launch Mechanical APDL.

With the GPU accelerator capability, the acceleration obtained by using the parallelism on the GPU hardware occurs only during the solution operations. Operational randomness and numerical round-off inherent to any parallel algorithm can cause slightly different results between runs on the same machine when using or not using the GPU hardware to accelerate the simulation.

The **ACCOPTION** command can also be used to control activation of the GPU accelerator capability.

3.2. Supported Analysis Types and Features

Some analysis types and features are not supported by the GPU accelerator capability. Supported functionality also depends on the specified GPU hardware. The following sections give general guidelines on what is and is not supported on various GPU hardware:

These are not comprehensive lists, but represent major features and capabilities found in the Mechanical APDL program.

3.2.1. nVIDIA GPU Hardware

This section lists analysis capabilities that are supported by the GPU accelerator capability when using nVIDIA GPU cards.

3.2.1.1. Supported Analysis Types

The following analysis types are supported and will use the GPU to accelerate the solution.

- Static linear or nonlinear analyses using the sparse, PCG, or JCG solver.
- Buckling analyses using the Block Lanczos or subspace eigensolver.
- Modal analyses using the Block Lanczos, subspace, PCG Lanczos, QR damped, unsymmetric, or damped eigensolver.
- Harmonic analyses using the full method and the sparse solver.
- Transient linear or nonlinear analyses using the full method and the sparse, PCG, or JCG solver.

In situations where the analysis type is not supported by the GPU accelerator capability, the solution will continue but GPU acceleration will not be used.

Shared-Memory Parallel Behavior

For the sparse solver (and eigensolvers based on the sparse solver), if one or more GPUs are requested, only a single GPU is used no matter how many are requested.

For the PCG and JCG solvers (and eigensolvers based on the PCG solver), all requested GPUs are used.

Distributed Memory Parallel Behavior

For the sparse solver (and eigensolvers based on the sparse solver), if the number of GPUs exceeds the number of processes (the `-na` value is greater than the `-np` value on the command line), the number of GPUs used equals the `-np` value. If the number of GPUs is less than the number of processes (`-na` is less than `-np`), all requested GPUs are used.

For the PCG and JCG solvers (and eigensolvers based on the PCG solver), if the number of GPUs exceeds the number of processes (`-na` is greater than `-np`), all requested GPUs are used. If the number of GPUs is less than the number of processes (`-na` is less than `-np`), all requested GPUs are used.

3.2.1.2. Supported Features

As the GPU accelerator capability currently only pertains to the equation solvers, virtually all features and element types are supported when using this capability with the supported equation solvers listed in Supported Analysis Types (p. 11). A few limitations exist and are listed below. In these situations, the solution will continue but GPU acceleration will not be used:

- Partial pivoting is activated when using the sparse solver. This most commonly occurs when using current technology elements with mixed u-P formulation or Lagrange multiplier based contact elements (TARGE169 through CONTA178).
- The memory saving option is activated (**MSAVE**,ON) when using the PCG solver.
- A non-supported equation solver is used (for example, ICCG, etc.).

3.2.2. Intel Xeon Phi Hardware

This section lists analysis capabilities that are supported by the GPU accelerator capability when using Intel Xeon Phi coprocessors.

Note that the Xeon Phi coprocessor has the following restrictions:

- You must be running on a Linux platform (Windows is not supported).
- Distributed-memory parallel processing is not supported.

3.2.2.1. Supported Analysis Types

The following analysis types are supported and will use the Xeon Phi coprocessor to accelerate the solution.

- Static linear or nonlinear analyses using the sparse solver (symmetric matrices only).
- Buckling analyses using the Block Lanczos or subspace eigensolver.
- Modal analyses using the Block Lanczos, subspace, or QR damped eigensolver.
- Harmonic analyses using the full method and the sparse solver (symmetric matrices only).
- Transient linear or nonlinear analyses using the full method and the sparse solver (symmetric matrices only)

In situations where the analysis type is not supported by the GPU accelerator capability, the solution will continue but GPU acceleration will not be used.

Shared-Memory Parallel Behavior

Only the sparse solver (and eigensolvers based on the sparse solver) can utilize Xeon Phi coprocessors. If one or more coprocessors are requested, all requested coprocessors are used.

3.2.2.2. Supported Features

As the GPU accelerator capability currently only pertains to the equation solvers, virtually all features and element types are supported when using this capability with the supported equation solvers listed

in Supported Analysis Types (p. 12). A few limitations exist and are listed below. In these situations, the solution will continue but GPU acceleration will not be used.

- Partial pivoting is activated when using the sparse solver. This most commonly occurs when using current technology elements with mixed u-P formulation or Lagrange multiplier based contact elements (TARGE169 through CONTA178).
- A non-supported equation solver is used (for example, PCG, ICCG, etc.).

3.3. Troubleshooting

This section describes problems which you may encounter while using the GPU accelerator capability, as well as methods for overcoming these problems. Some of these problems are specific to a particular system, as noted.

Note that GPU acceleration is not supported on Windows 32-bit platforms.

NVIDIA GPUs support various compute modes (for example, Exclusive thread, Exclusive process). Only the default compute mode is supported. Using other compute modes may cause the program to fail to launch.

To list the GPU devices installed on the machine, set the **ANSGPU_PRINTDEVICES** environment variable to a value of 1. The printed list may or may not include graphics cards used for display purposes, along with any graphics cards used to accelerate your simulation.

No Supported Devices

Be sure that a supported GPU device is properly installed and configured. Check the driver level to be sure it is current or newer than the driver version supported for your particular device. (See the GPU requirements outlined in the Windows Installation Guide and the Linux Installation Guide.)

Note

On Windows, the use of Remote Desktop may disable the use of a GPU device. Launching Mechanical APDL through the ANSYS Remote Solve Manager (RSM) when RSM is installed as a service may also disable the use of a GPU. In these two scenarios, the GPU Accelerator Capability cannot be used. Using the TCC (Tesla Compute Cluster) driver mode, if applicable, can circumvent this restriction.

No Valid Devices

A GPU device was detected, but it is not a supported GPU device. Be sure that a supported GPU device is properly installed and configured. Check the driver level to be sure it is current or newer than the supported driver version for your particular device. (See the GPU requirements outlined in the Windows Installation Guide and the Linux Installation Guide.)

Poor Acceleration or No Acceleration

Simulation includes non-supported features A GPU device will only accelerate certain portions of a simulation, mainly the solution time. If the bulk of the simulation time is spent outside of solution, the GPU cannot have a significant impact on the overall analysis time. Even if the bulk of the simulation is spent inside solution, you must be sure that a supported equation solver is utilized during solution and that no unsupported options are used. Messages are printed in the output to alert users when a GPU is being used, as well as when unsupported options/features are chosen which deactivate the GPU accelerator capability.

Simulation has too few DOF (degrees of freedom) Some analyses (such as transient analyses) may require long compute times, not because the number of DOF is large, but because a large number of calculations are performed (i.e., a very large number of time steps). Generally, if the number of DOF is relatively small, GPU acceleration will not significantly decrease the solution time. Consequently, for small models with many time steps, GPU acceleration may be poor because the model size is too small to fully utilize a GPU.

Simulation does not fully utilize the GPU Only simulations that spend a lot of time performing calculations that are supported on a GPU can expect to see significant speedups when a GPU is used. Only certain computations are supported for GPU acceleration. Therefore, users should check to ensure that a high percentage of the solution time was spent performing computations that could possibly be accelerated on a GPU. This can be done by reviewing the equation solver statistics files as described below. See Measuring ANSYS Performance in the *Performance Guide* for more details on the equation solver statistics files.

- **PCG solver file:** The .PCS file contains statistics for the PCG iterative solver. You should first check to make sure that the GPU was utilized by the solver. This can be done by looking at the line which begins with: "Number of cores used". The string "GPU acceleration enabled" will be added to this line if the GPU hardware was used by the solver. If this string is missing, the GPU was not used for that call to the solver. Next, you should study the elapsed times for both the "Preconditioner Factoring" and "Multiply With A22" computations. GPU hardware is only used to accelerate these two sets of computations. The wall clock (or elapsed) times for these computations are the areas of interest when determining how much GPU acceleration is achieved.

- **Sparse solver files:** The .BCS (or .DSP) file contains statistics for the sparse direct solver. You should first check to make sure that the GPU was utilized by the solver. This can be done by looking for the following line: "GPU acceleration activated". This line will be printed if the GPU hardware was used. If this line is missing, the GPU was not used for that call to the solver. Next, you should check the percentage of factorization computations (flops) which were accelerated on a GPU. This is shown by the line: "percentage of GPU accelerated flops". Also, you should look at the time to perform the matrix factorization, shown by the line: "time (cpu & wall) for numeric factor". GPU hardware is only used to accelerate the matrix factor computations. These lines provide some indication of how much GPU acceleration is achieved.

- **Eigensolver files:** The .BCS file is written for the Block Lanczos eigensolver and can be used as described above for the sparse direct solver. The .PCS file is written for the PCG Lanczos eigensolver and can be used as described above for the PCG iterative solver.

Using multiple GPU devices When using the sparse solver in a shared-memory parallel solution, it is expected that running a simulation with multiple GPU devices will not improve performance compared to running with a single GPU device. In a shared-memory parallel solution, the sparse solver can only make use of one GPU device.

Oversubscribing GPU hardware The program automatically determines which GPU devices to use. In a multiuser environment, this could mean that one or more of the same GPUs are picked when multiple simulations are run simultaneously, thus oversubscribing the hardware.

- If only a single GPU accelerator device exists in the machine, then only a single user should attempt to make use of it, much in the same way users should avoid oversubscribing their CPU cores.

- If multiple GPU accelerator devices exist in the machine, you can set the **ANSGPU_DEVICE** environment variable, in conjunction with the **ANSGPU_PRINTDEVICES** environment variable mentioned above, to specify which particular GPU accelerator devices to use during the solution.

For example, consider a scenario where **ANSGPU_PRINTDEVICES** shows that four GPU devices are available with device ID values of 1, 3, 5, and 7 respectively, and only the second and third devices are supported for GPU acceleration. To select only the second supported GPU device, set **ANSGPU_DEVICE** = 5. To select the first and second supported GPU devices, set **ANSGPU_DEVICE** = 3:5.

Chapter 4: Using Distributed ANSYS

When running a simulation, the solution time is typically dominated by three main parts: the time spent to create the element matrices and form the global matrices or global systems of equations, the time to solve the linear system of equations, and the time spent calculating derived quantities (such as stress and strain) and other requested results for each element.

The distributed-memory parallelism offered via Distributed ANSYS allows the entire solution phase to run in parallel, including the stiffness matrix generation, linear equation solving, and results calculations. As a result, a simulation using distributed-memory parallel processing usually achieves much faster solution times than a similar run performed using shared-memory parallel processing, particularly at higher core counts.

Distributed ANSYS can run a solution over multiple cores on a single machine or on multiple machines (that is, a cluster). It automatically decomposes the model into smaller domains, transfers the domains to each core, solves each domain simultaneously, and creates a complete solution to the model. The memory and disk space required to complete the solution can also be distributed over multiple machines. By utilizing all of the resources of a cluster (computing power, RAM, memory and I/O bandwidth), distributed-memory parallel processing can be used to solve very large problems much more efficiently compared to the same simulation run on a single machine.

Distributed ANSYS Behavior

Distributed ANSYS works by launching multiple ANSYS processes on either a single machine or on multiple machines (as specified by one of the following command line options: -np, -machines, or -mpifile). The machine that the distributed run is launched from is referred to as the master or host machine (or in some cases, primary compute node), and the other machines are referred to as the slave machines (or compute nodes). The first process launched on the master machine is referred to as the master or host process; all other processes are referred to as the slave processes.

Each Distributed ANSYS process is essentially a running process of shared-memory ANSYS. These processes are launched through the specified MPI software layer. The MPI software allows each Distributed ANSYS process to communicate, or exchange data, with the other processes involved in the distributed simulation.

Distributed ANSYS does not currently support all of the analysis types, elements, solution options, etc. that are available with shared-memory ANSYS (see Supported Features (p. 33)). In some cases, Distributed ANSYS stops the analysis to avoid performing an unsupported action. If this occurs, you must launch shared-memory ANSYS to perform the simulation. In other cases, Distributed ANSYS will automatically disable the distributed-memory parallel processing capability and perform the operation using shared-memory parallelism. This disabling of the distributed-memory parallel processing can happen at various levels in the program.

The master process handles the inputting of commands as well as all of the pre- and postprocessing actions. Only certain commands (for example, the **SOLVE** command and supporting commands such as **/SOLU**, **FINISH**, **/EOF**, **/EXIT**, and so on) are communicated to the slave processes for execution. Therefore, outside of the SOLUTION processor (**/SOLU**), Distributed ANSYS behaves very similar to shared-memory ANSYS. The master process works on the entire model during these pre- and postpro-

cessing steps and may use shared-memory parallelism to improve performance of these operations. During this time, the slave processes wait to receive new commands from the master process.

Once the **SOLVE** command is issued, it is communicated to the slave processes and all Distributed ANSYS processes become active. At this time, the program makes a decision as to which mode to use when computing the solution. In some cases, the solution will proceed using only a distributed-memory parallel (DMP) mode. In other cases, similar to pre- and postprocessing, the solution will proceed using only a shared-memory parallel (SMP) mode. In a few cases, a mixed mode may be implemented which tries to use as much distributed-memory parallelism as possible for maximum performance. These three modes are described further below.

Pure DMP mode The simulation is fully supported by Distributed ANSYS, and distributed-memory parallelism is used throughout the solution. This mode typically provides optimal performance in Distributed ANSYS.

Mixed mode The simulation involves an equation solver that is not supported by Distributed ANSYS. In this case, distributed-memory parallelism is used throughout the solution, except for the equation solver. When the equation solver is reached, the slave processes in Distributed ANSYS simply wait while the master process uses shared-memory parallelism to compute the equation solution. After the equation solution is computed, the slave processes continue to compute again until the entire solution is completed.

Pure SMP mode The simulation involves an analysis type or feature that is not supported by Distributed ANSYS. In this case, distributed-memory parallelism is disabled at the onset of the solution, and shared-memory parallelism is used instead. The slave processes in Distributed ANSYS are not involved at all in the solution but simply wait while the master process uses shared-memory parallelism to compute the entire solution.

When using shared-memory parallelism inside of Distributed ANSYS (in mixed mode or SMP mode, including all pre- and postprocessing operations), the master process will not use more cores on the master machine than the total cores you specify to be used for the Distributed ANSYS solution. This is done to avoid exceeding the requested CPU resources or the requested number of licenses.

The following table shows which steps, including specific equation solvers, can be run in parallel using shared-memory ANSYS and Distributed ANSYS.

Table 4.1: Parallel Capability in Shared-Memory and Distributed ANSYS

Solvers/Feature	Shared-Memory ANSYS	Distributed ANSYS
Sparse	Y	Y
PCG	Y	Y
ICCG	Y	Y [1]
JCG	Y	Y [1] [2]
QMR	Y	Y [1]
Block Lanczos eigensolver	Y	Y [1]
PCG Lanczos eigensolver	Y	Y
Supernode eigensolver	Y	Y [1]
Subspace eigensolver	Y	Y
Unsymmetric eigensolver	Y	Y

Solvers/Feature	Shared-Memory ANSYS	Distributed ANSYS
Damped eigensolver	Y	Y
QRDAMP eigensolver	Y	Y [3]
Element formulation, results calculation	Y	Y
Graphics and other pre- and postprocessing	Y	Y [1]

1. This solver/operation only runs in mixed mode.

2. For static analyses and transient analyses using the full method (**TRNOPT**,FULL), the JCG equation solver runs in pure DMP mode only when the matrix is symmetric. Otherwise, it runs in SMP mode.

3. The QRDAMP eigensolver only runs in pure SMP mode.

The maximum number of cores allowed in a Distributed ANSYS analysis is currently set at 8192. Therefore, you can run Distributed ANSYS using anywhere from 2 to 8192 cores (assuming the appropriate HPC licenses are available) for each individual job. Performance results vary widely for every model when using any form of parallel processing. For every model, there is a point where using more cores does not significantly reduce the overall solution time. Therefore, it is expected that most models run in Distributed ANSYS can not efficiently make use of hundreds or thousands of cores.

Files generated by Distributed ANSYS are named `Jobname`*n*`.ext`, where *n* is the process number. (See Differences in General Behavior (p. 35) for more information.) The master process is always numbered 0, and the slave processes are 1, 2, etc. When the solution is complete and you issue the **FINISH** command in the SOLUTION processor, Distributed ANSYS combines all `Jobname`*n*`.RST` files into a single `Jobname.RST` file, located on the master machine. Other files, such as `.MODE`, `.ESAV`, `.EMAT`, etc., may be combined as well upon finishing a distributed solution. (See Differences in Postprocessing (p. 38) for more information.)

The remaining sections explain how to configure your environment to run Distributed ANSYS, how to run a Distributed ANSYS analysis, and what features and analysis types are supported in Distributed ANSYS. You should read these sections carefully and fully understand the process before attempting to run a distributed analysis. The proper configuration of your environment and the installation and configuration of the appropriate MPI software are critical to successfully running a distributed analysis.

4.1. Configuring Distributed ANSYS

Before running an analysis with Distributed ANSYS, you must configure your system properly. Your system must meet specified criteria and you must have supported MPI software correctly installed.

The following topics are available for configuring and starting Distributed ANSYS:

4.1.1. Prerequisites for Running Distributed ANSYS

Whether you are running on a single machine or multiple machines, the following condition is true:

- Distributed ANSYS allows you to use two cores without using any HPC licenses. Additional licenses will be needed to run a distributed solution with more than two cores. Several HPC license options are available. For more information, see HPC Licensing in the *Parallel Processing Guide*.

If you are running on a single machine, there are no additional requirements for running a distributed solution.

If you are running across multiple machines (e.g., a cluster), your system must meet these additional requirements to run a distributed solution.

- Homogeneous network: All machines in the cluster must be the same type, OS level, chip set, and interconnects.

- You must be able to remotely log in to all machines, and all machines in the cluster must have identical directory structures (including the ANSYS 15.0 installation, MPI installation, and on some systems, working directories). Do not change or rename directories after you've launched ANSYS. For more information on files that are written and their location, see Controlling Files that Distributed ANSYS Writes in the *Parallel Processing Guide*.

- All machines in the cluster must have ANSYS 15.0 installed, or must have an NFS mount to the ANSYS 15.0 installation. If not installed on a shared file system, ANSYS 15.0 must be installed in the same directory path on all systems.

- All machines must have the same version of MPI software installed and running. The table below shows the MPI software and version level supported for each platform. For Linux platforms, the MPI software is included with the ANSYS 15.0 installation. For Windows platforms, you must install the MPI software as described later in this document.

4.1.1.1. MPI Software

The MPI software supported by Distributed ANSYS depends on the platform. The following table lists the type of MPI software supported for each platform. Platform MPI and Intel MPI are included on the Linux installation media and are installed automatically when you install ANSYS 15.0. Instructions for installing the MPI software on Windows systems can be found later in this document (see Installing the Software).

Distributed ANSYS runs on the following platforms:

Intel Xeon EM64T 64-bit Linux (Platform MPI, Intel MPI)
AMD Opteron 64-bit Linux (Platform MPI, Intel MPI)
Windows 32-bit (Platform MPI, Intel MPI)
Windows 64-bit (Platform MPI, MS MPI, Intel MPI)
Windows HPC Server 2008 x64 (Microsoft HPC Pack (MS MPI))

Table 4.2: Platforms and MPI Software

Platform	MPI Software	More Information
Linux, Intel Xeon EM64T and AMD Opteron 64-bit	Platform MPI 9.1 Intel MPI 4.1	Platform MPI: http://www-03.ibm.com/systems/technicalcomputing/platformcomputing/products/mpi/index.html Intel MPI: http://software.intel.com/en-us/articles/intel-mpi-library-documentation/

Platform	MPI Software	More Information
Windows 32-bit / Windows XP / Windows 7 / Windows 8 Windows 64-bit / Windows XP x64 / Windows 7 x64 / Windows 8 x64	Platform MPI 9.1 Intel MPI 4.1	Platform MPI: http://www-03.ibm.com/systems/technicalcomputing/platformcomputing/products/mpi/index.html Intel MPI: http://software.intel.com/en-us/articles/intel-mpi-library-documentation/
Windows HPC Server 2008 x64	Microsoft HPC Pack (MS MPI)	http://www.microsoft.com/hpc/

ANSYS LS-DYNA If you are running ANSYS LS-DYNA, you can use LS-DYNA's parallel processing (MPP or SMP) capabilities. Use the launcher or the command line method as described in *Activating Distributed ANSYS* in the *Parallel Processing Guide* to run LS-DYNA MPP. For Windows and Linux systems, please see the following table for LS-DYNA MPP MPI support. For more information on using in ANSYS LS-DYNA general, and its parallel processing capabilities specifically, see the *ANSYS LS-DYNA User's Guide*.

Table 4.3: LS-DYNA MPP MPI Support on Windows and Linux

MPI version for DYNA MPP	32-bit Windows	64-bit Windows	64-bit Linux
Platform MPI	n/a	X	X
MS MPI	n/a	X	n/a

4.1.1.2. Installing the Software

To run Distributed ANSYS on a cluster, you must install ANSYS 15.0 on all machines in the cluster, or have an NFS mount to the ANSYS 15.0 installation. Install ANSYS 15.0 following the instructions in the *ANSYS, Inc. Installation Guide* for your platform. Be sure to complete the installation, including all required post-installation procedures. On Windows systems, you must use the Universal Naming Convention (UNC) for all file and path names for Distributed ANSYS to work correctly.

Installing Platform MPI on Windows

You can install Platform MPI from the installation launcher by choosing **Install MPI for ANSYS, Inc. Parallel Processing**. On the following screen, choose to install Platform MPI. The Platform MPI installation program will start. A Platform MPI installation README file will open simultaneously. Follow the instructions in the README file as you complete the Platform MPI installation.

The instructions for installing Platform MPI are also found in the installation directory in the following README files:

```
Program Files\Ansys Inc\V150\commonfiles\MPI\Platform\9.1\Windows\INSTALL_PLATFORM-MPI_README.mht
```

or

```
Program Files\Ansys Inc\V150\commonfiles\MPI\Platform\9.1\Windows\INSTALL_PLATFORM-MPI_README.docx
```

Installing Intel MPI on Windows

You can install Intel MPI from the Installation launcher by choosing **Install MPI for ANSYS, Inc. Parallel Processing**. On the following screen, choose to install Intel MPI. The Intel MPI installation program will start. An Intel MPI installation README file will open simultaneously. Follow the instructions in the README file as you complete the Intel MPI installation.

The instructions for installing Intel MPI are also found in the installation directory in the following README files:

```
Program Files\Ansys Inc\V150\commonfiles\MPI\Intel\4.1\Windows\INSTALL_IN-
TEL-MPI_README.mht
```

or

```
Program Files\Ansys Inc\V150\commonfiles\MPI\Intel\4.1\Windows\INSTALL_IN-
TEL-MPI_README.docx
```

Microsoft HPC Pack (Windows HPC Server 2008)

You must complete certain post-installation steps before running Distributed ANSYS on a Microsoft HPC Server 2008 system. The post-installation instructions provided below assume that Microsoft HPC Server 2008 and Microsoft HPC Pack (which includes MS MPI) are already installed on your system. The post-installation instructions can be found in the following README files:

```
Program Files\Ansys Inc\V150\commonfiles\MPI\WindowsHPC\README.mht
```

or

```
Program Files\Ansys Inc\V150\commonfiles\MPI\WindowsHPC\README.docx
```

Microsoft HPC Pack examples are also located in `Program Files\Ansys Inc\V150\common-files\MPI\WindowsHPC`. Jobs are submitted to the Microsoft HPC Job Manager either from the command line or the Job Manager GUI.

To submit a job via the GUI, go to **Start> All Programs> Microsoft HPC Pack> HPC Job Manager**. Then click on **Create New Job from Description File**.

4.1.2. Setting Up the Cluster Environment for Distributed ANSYS

After you've ensured that your cluster meets the prerequisites and you have ANSYS 15.0 and the correct version of MPI installed, you need to configure your distributed environment using the following procedure.

1. Obtain the machine name for each machine on the cluster.

   Windows XP or Windows 7:
   Right-click on **My Computer**, left-click on **Properties**, and select the **Network Identification** or **Computer Name** tab. The full computer name will be listed. Note the name of each machine (not including the domain).

   Windows 8:
   From the **All apps** screen, open the **Control Panel**. Click **System**and **Security> System**. The full computer name will be listed. Note the name of each machine (not including the domain).

Linux:

Type **hostname** on each machine in the cluster. Note the name of each machine. You will need this name to set up the .rhosts file, as well as for the **ANS_ADMIN** utility.

2. **Linux only:** Set up the .rhosts file on each machine. The .rhosts file lists all machines in the cluster. The machines should be listed using their complete system name, as taken from **hostname**. For example, an .rhosts file for a two-machine cluster might look like this:

```
golinux1.ansys.com jqd
golinux2 jqd
```

Change/verify .rhosts file permissions on all machines by issuing:

```
chmod 600 .rhosts
```

Verify communication between machines via rsh or ssh (e.g., **rsh golinux2 ls**). You should not be prompted for a password. If you are, check the .rhosts permissions and machine names for correctness. For more information on using remote shells, see the man pages for rsh or ssh.

3. If you want the list of machines to be populated in the Mechanical APDL Product Launcher, you need to configure the hosts150.ans file. You can use the **ANS_ADMIN** utility to configure this file. You can manually modify the file later, but we strongly recommend that you use **ANS_ADMIN** to create this file initially to ensure that you establish the correct format.

Windows XP or Windows 7:

Start >Programs >ANSYS 15.0 >Utilities >ANS_ADMIN 15.0

Windows 8:

All apps >ANSYS 15.0 > ANS_ADMIN 15.0

Linux:

/ansys_inc/v150/ansys/bin/ans_admin150

The **ANS_ADMIN** interface is similar for Windows and Linux:

- Choose **ANSYS/Workbench Configure Cluster** to configure the hosts150.ans file.
- Under **Select file to configure**, choose the hosts150.ans file to be configured and choose **Configure for Distributed ANSYS**. Click **OK**.
- Enter the system name (from Step 1) in the **Machine hostname** field and click **Add**. On the next dialog box, enter the system type in the **Machine type** drop-down, and the number of cores in the **Max number of jobs/processors** field. Click **Add**. Repeat this step for each machine in the cluster.
- When you are finished adding machines, click **Close** then **File > Exit**.

The hosts150.ans file should be located in your current working directory, your home directory, or the apdl directory. All three locations are equally valid; the location should be chosen based on user preference.

4. **Windows only:** Verify that all required environment variables are properly set. If you followed the post-installation instructions described above for Microsoft HPC Pack (Windows HPC Server 2008), these variable should be set automatically.

On the head node, where ANSYS 15.0 is installed, check these variables:

ANSYS150_DIR=C:\Program Files\ANSYS Inc\v150\ansys

ANSYSLIC_DIR=C:\Program Files\ANSYS Inc\Shared Files\Licensing

where `C:\Program Files\ANSYS Inc` is the location of the product install and `C:\Program Files\ANSYS Inc\Shared Files\Licensing` is the location of the licensing install. If your installation locations are different than these, specify those paths instead.

On Windows systems, you must use the Universal Naming Convention (UNC) for all ANSYS, Inc. environment variables on the compute nodes for Distributed ANSYS to work correctly.

On the compute nodes, check these variables:

ANSYS150_DIR=*head_node_machine_name*\ANSYS Inc\v150\ansys

ANSYSLIC_DIR=*head_node_machine_name*\ANSYS Inc\Shared Files\Licensing

For distributed LS-DYNA:

On the head node and the compute nodes, set **LSTC_LICENSE** to ANSYS. This tells the LS-DYNA executable to use ANSYS, Inc. licensing.

Since the LS-DYNA run will use ANSYS, Inc. licensing for LS-DYNA, you do not need to set **LSTC_LICENSE_SERVER**.

5. **Windows only:** Share out the ANSYS `Inc` directory on the head node with full permissions so that the compute nodes can access it.

4.1.2.1. Optional Setup Tasks

The tasks explained in this section are optional. They are not required to get Distributed ANSYS to run correctly, but they may be useful for achieving the most usability and efficiency, depending on your system configuration.

On Linux systems, you can also set the following environment variables:

- **ANSYS_NETWORK_START** - This is the time, in seconds, to wait before timing out on the start-up of the client (default is 15 seconds).
- **ANSYS_NETWORK_COMM** - This is the time to wait, in seconds, before timing out while communicating with the client machine (default is 5 seconds).
- **ANS_SEE_RUN_COMMAND** - Set this environment variable to 1 to display the actual mpirun command issued from ANSYS.

On Linux systems running Platform MPI:

- **MPI_REMSH** - This is the path to the remote shell (ssh or rsh). Set this environment variable to specify a full path to a remote shell. For example, setting **`MPI_REMSH = /usr/bin/ssh`** will use ssh instead of the default remote shell (rsh). Note that selecting the **Use Secure Shell instead of Remote Shell** option on the launcher will override **MPI_REMSH**, if **MPI_REMSH** is not set or is set to a different location. You can also issue the `-usessh` command line option to use ssh instead of rsh. The command line option will override the environment variable setting as well.

- **MPI_WORKDIR** - Set this environment variable to specify a working directory on either the master and all nodes, or on specific nodes individually. For more information, see Controlling Files that Distributed ANSYS Writes.
- **MPI_IC_ORDER** - Set this environment variable to specify the order in which the interconnects on the system are to be used. The interconnects will be tried in the order listed from left to right. If an interconnect is listed in uppercase, no interconnects listed after that one will be tried. If **MPI_IC_ORDER** is not set, the fastest interconnect available on the system is used. See the Platform MPI documentation for more details.
- **MPI_ICLIB_<*interconnect*>** - Set this environment variable to the interconnect location if the interconnect is not installed in the default location:

  ```
  setenv MPI_ICLIB_GM <path>/lib64/libgm.so
  ```

  See the Platform MPI documentation for the specific interconnect names (e.g., **MPI_ICLIB_GM**).
- **MPIRUN_OPTIONS** - Set this environment variable to -prot to display a grid of interconnects among the systems being used for distributed processing.

On Linux systems running Intel MPI:

- Issue the command line option -usessh to use ssh instead of rsh.
- See the Intel MPI reference manual (for Linux) for further information and additional environment variables and their settings: http://software.intel.com/en-us/articles/intel-mpi-library-documentation/.

To verify that these environment variables are set correctly on each machine, run:

```
rsh machine1 env
```

On Windows systems, you can set the following environment variables to display the actual mpirun command issued from ANSYS:

- **ANS_SEE_RUN** = TRUE
- **ANS_CMD_NODIAG** = TRUE

4.1.2.2. Using the mpitest Program

The `mpitest` program performs a simple communication test to verify that the MPI software is set up correctly. The `mpitest` program should start without errors. If it does not, check your paths, `.rhosts` file, and permissions; correct any errors, and rerun.

When running the mpitest program, you must use an even number of nodes.

On Linux:

For Platform MPI (default), issue the following command:

```
mpitest150 -machines machine1:2
```

For Intel MPI, issue the following command:

```
mpitest150 -mpi intelmpi -machines machine1:2
```

You can use any of the same command line arguments (such as `-machines`) with the `mpitest` program as you can with Distributed ANSYS.

On Windows:

Issue the following command to run a local test on Windows using Platform MPI:

```
ansys150 -np 2 -mpitest
```

Use the following procedure to run a distributed test on Windows using Platform MPI:

1. Create a file named `machines` in your local/home directory. Open the `machines` file in an editor.

2. Add your master and slave machines in your cluster. For example, in this cluster of two machines, the master machine is **`gowindows1`**. List the machine name separately for each core on that machine. For example, if **`gowindows1`** has four cores and **`gowindows2`** has two, the `machines` file would look like this:

   ```
   gowindows1
   gowindows1
   gowindows1
   gowindows1
   gowindows2
   gowindows2
   ```

3. From a command prompt, navigate to your working directory. Run the following:

   ```
   ansys150 -mpifile machines -mpitest
   ```

4.1.2.3. Interconnect Configuration

Low-end hardware, such as slow interconnects, will reduce the speed improvements you see in a distributed analysis. For optimal performance, we typically recommend that you use an interconnect with a communication speed of 1000 megabytes/second or higher.

Distributed ANSYS supports the following interconnects. Not all interconnects are available on all platforms; see http://www.ansys.com/Support/Platform+Support for a current list of supported interconnects. Other interconnects may work but have not been tested.

- InfiniBand (recommended)
- Myrinet
- GigE
- Ethernet (not recommended)

Interconnects plug into a PCI (Peripheral Component Interconnect), PCI-X (extended), or PCIe (PCI Express) slot on the system. You will need a PCI-X or a PCIe slot for the faster interconnects to accommodate the higher speeds.

Hardware for specific types of interconnects is generally incompatible with other proprietary interconnect types (except Ethernet and GiGE).

Systems can have a network of several different types of interconnects. Each interconnect must be assigned a unique hostname and IP address.

On Windows x64 systems, use the Network Wizard in the Compute Cluster Administrator to configure your interconnects. See the Compute Cluster Pack documentation for specific details on setting up the

interconnects. You may need to ensure that Windows Firewall is disabled for Distributed ANSYS to work correctly.

4.2. Activating Distributed ANSYS

After you've completed the configuration steps, you can use several methods to start Distributed ANSYS: We recommend that you use the Mechanical APDL Product Launcher to ensure the correct settings. All methods are explained here.

- Use the launcher
- Use the command line
- Use the HPC Job Manager on Windows x64 systems to run across multiple machines
- Use Remote Solve in ANSYS Workbench.

Notes on Running Distributed ANSYS:

You can use an NFS mount to the ANSYS 15.0 installation; however, we do not recommend NFS-mounting the working directories. Doing so can result in significant declines in performance.

Only the master process reads the config150.ans file. Distributed ANSYS ignores the **/CONFIG**,NOELDB, **/CONFIG**,FSPLIT, and **/CONFIG**,NORSTGM commands.

The program limits the number of processes used to be less than or equal to the number of physical cores on the machine. This is done to avoid running the program on virtual cores (for example, by means of hyperthreading), which typically results in poor per-core performance. For optimal performance, consider closing down all other applications before launching Mechanical APDL.

4.2.1. Starting Distributed ANSYS via the Launcher

Use the following procedure to start Distributed ANSYS via the launcher.

This procedure is also used for running ANSYS LS-DYNA MPP. See LS-DYNA Parallel Processing Capabilities in the *ANSYS LS-DYNA User's Guide* for a detailed explanation on using ANSYS LS-DYNA's parallel processing capabilities.

1. Open the Mechanical APDL Product Launcher:

   Windows:
   Start >Programs >ANSYS 15.0 >Mechanical APDL Product Launcher 15.0

   Linux:

   ```
   launcher150
   ```

2. Select the correct environment and license.
3. Go to the **High Performance Computing Setup** tab. Select **Use Distributed Computing (MPP)**.

   Specify the MPI type to be used for this distributed run. MPI types include:

   - PCMPI (Platform MPI)

- Intel MPI
- MS MPI (Windows 64-bit only)

See Table 4.2: Platforms and MPI Software (p. 20) for the specific MPI version for each platform. If you choose MS MPI, you cannot specify multiple hosts or an MPI file.

Choose whether you want to run on a local machine, specify multiple hosts, or specify an existing MPI file (such as a `host.list` or a Platform MPI appfile):

- If local machine, specify the number of cores you want to use on that machine.
- If multiple hosts, select the machines you want to use from the list of available hosts. The list of available hosts is populated from the `hosts150.ans` file. Click on the machines you want to use and click **Add** to move them to the **Selected Hosts** list to use them for this run. If you click **Add** more than once for a machine, the number of cores to be used on that machine will increment each time, up to the maximum allowed in the `hosts150.ans` file. (Note that the **Select Multiple Hosts** option is not available when running the LS-DYNA MPP version on a Windows system.)

  You can also add or remove a host, but be aware that adding or removing a host from here will modify *only this run*; the `hosts150.ans` file will *not be updated* with any new information from this dialog box.
- If specifying an MPI file, type in the full path to the file, or browse to the file. If typing in the path, you must use the absolute path.

Additional Options for Linux systems using Platform MPI On these systems, you can choose to use secure shell (SSH) instead of remote shell (RSH). This option will override **MPI_REMSH**, if the path to SSH is different. See Optional Setup Tasks (p. 24) for more information on **MPI_REMSH**. ANSYS uses RSH as the default, whereas Platform MPI uses SSH as the default.

If you are using the launcher, you can select the **Use launcher-specified working directory on all nodes** option on the **High Performance Computing Setup** tab. This option uses the working directory as specified on the **File Management** tab as the directory structure on the master and all nodes. If you select this option, all machines will require the identical directory structure matching the working directory specified on the launcher. This option will override any existing **MPI_WORKDIR** settings on the master or the nodes.

4. Click **Run** to launch ANSYS.

4.2.2. Starting Distributed ANSYS via Command Line

You can also start Distributed ANSYS via the command line using the following procedures.

These procedures are also used for running ANSYS LS-DYNA MPP. To run the LS-DYNA MPP version, substitute **`lsdyna150`** for **`ansys150`** in the examples below. Only the command line options specific to Distributed ANSYS apply to LS-DYNA MPP (i.e., `-dis`, `-np`, `-machines`, `-mpifile`). See LS-DYNA Parallel Processing Capabilities in the *ANSYS LS-DYNA User's Guide* for a detailed explanation on using ANSYS LS-DYNA's parallel processing capabilities.

Running on a Local Host If you are running Distributed ANSYS locally (that is, running across multiple cores on a single machine), you need to specify the number of cores you want to use:

```
ansys150 -dis -np n
```

You may also need to specify the MPI software using the `-mpi` command line option:

- `-mpi pcmpi` : Platform MPI (default)
- `-mpi intelmpi` : Intel MPI

If you are using Platform MPI, you do not need to specify the MPI software via the command line option. To specify Intel MPI, use the `-mpi intelmpi` command line option as shown below:

```
ansys150 -dis -mpi intelmpi -np n
```

For example, if you run a job in batch mode on a local host using four cores with an input file named `input1` and an output file named `output1`, the launch commands for Linux and Windows would be as shown below.

On Linux:

```
ansys150 -dis -np 4 -b < input1 > output1  (for default Platform MPI)
```

or

```
ansys150 -dis -mpi intelmpi -np 4 -b < input1 > output1  (for Intel MPI)
```

On Windows:

```
ansys150 -dis -np 4 -b -i input1 -o output1  (for default Platform MPI)
```

or

```
ansys150 -dis -mpi intelmpi -np 4 -b -i input1 -o output1  (for Intel MPI)
```

Running on Multiple Hosts If you are running Distributed ANSYS across multiple hosts, you need to specify the number of cores you want to use on each machine:

```
ansys150 -dis -machines machine1:np:machine2:np:machine3:np
```

You may also need to specify the MPI software. The default is Platform MPI. To specify Intel MPI, use the -mpi command line option as shown below:

```
ansys150 -dis -mpi intelmpi -machines machine1:np:machine2:np:machine3:np
```

For example, if you run a job in batch mode using two machines (using four cores on one machine and using two cores on the other machine), with an input file named `input1` and an output file named `output1`, the launch commands for Linux and Windows would be as shown below.

On Linux:

```
ansys150 -dis -b -machines machine1:4:machine2:2 < input1 > output1  (for default Platform MPI)
```

or

```
ansys150 -dis -mpi intelmpi -b -machines machine1:4:machine2:2 < input1 > output1 (for Intel MPI)
```

On Windows:

```
ansys150 -dis -b -machines machine1:4:machine2:2 -i input1 -o output1  (for default Platform MPI)
```

or

```
ansys150 -dis -mpi intelmpi -b -machines machine1:4:machine2:2 -i input1 -o output1  (for Intel MPI)
```

The first machine specified with `-machines` in a Distributed ANSYS run must be the host machine and must contain any files necessary for the initiation of your job (i.e., input file, config150.ans file, database file, etc.).

If both the `-np` and `-machines` options are used on a single command line, the `-np` will be ignored.

Note that the `-machines` option is not available when running the LS-DYNA MPP version on a Windows system.

4.2.3. Starting Distributed ANSYS via the HPC Job Manager

If you are running on Windows x64 systems using Microsoft HPC Pack (MS MPI), you need to use the HPC Job Manager to start Distributed ANSYS. For more information, refer to the following README files:

```
Program Files\Ansys Inc\V150\commonfiles\MPI\WindowsHPC\README.mht
```

or

```
Program Files\Ansys Inc\V150\commonfiles\MPI\WindowsHPC\README.docx
```

4.2.4. Starting Distributed ANSYS in ANSYS Workbench

If you are running ANSYS Workbench, you can start a Distributed ANSYS job in the Mechanical application; go to **Tools > Solve Process Settings**. Select the remote solve process you want to use and click the **Advanced** button. You will need to select **Distribute ANSYS Solution (if possible)** and enter the command line arguments to be submitted. Use the options as described in Starting Distributed ANSYS via Command Line (p. 28). If **Distribute ANSYS Solution (if possible)** is selected, you will not need to specify the `-dis` flag on the command line.

If you are running a remote solution on multiple machines, use the `-machines` option to specify the machines and cores on which to run the job. If you are running a remote solution on one machine with multiple cores, specify the number of cores in the **Max Number of Utilized Processors** field. You will not need to add any command line arguments.

For more information on running a distributed solution in ANSYS Workbench, see Using Solve Process Settings.

4.2.5. Using MPI appfiles

You can specify an existing MPI file (such as a Platform MPI appfile) on the command line, rather than typing out multiple hosts or a complicated command line:

```
ansys150 -dis -mpifile appfile_name
```

For an Intel MPI appfile, include the -mpi command line option:

```
ansys150 -dis -mpi intelmpi -mpifile appfile_name
```

The format of the appfile is system-dependent.

If the file is not in the current working directory, you will need to include the full path to the file. The file must reside on the local machine.

You cannot use the `-mpifile` option in conjunction with the `-np` (local host) or `-machines` (multiple hosts) options.

If the **Specify Multiple Hosts** launcher option or the `-machines` command line option was used, ANSYS generates a default appfile named `host.list`. You can rename this file, move it, or modify it for future runs if necessary. See the documentation for your vendor/MPI type for details on working with the appfile.

Using the Platform MPI appfile Platform MPI uses an appfile to define the machines in the array (or the local host). A typical Platform MPI appfile might look like this:

```
-h mach1 -np 2 /ansys_inc/v150/ansys/bin/ansysdis150 -dis
-h mach2 -np 2 /ansys_inc/v150/ansys/bin/ansysdis150 -dis
```

See the Platform MPI user documentation for details on working with the Platform MPI appfile.

Using the Intel MPI appfile Intel MPI uses an appfile to define the machines in the array (or the local host). A typical Intel MPI appfile might look like this:

```
-host mach1  -env env_vars env_var_settings -np 2 /ansys_inc/v150/ansys/bin/ansysdis150 -dis -mpi INTELMPI
-host mach2  -env env_vars env_var_settings -np 2 /ansys_inc/v150/ansys/bin/ansysdis150 -dis -mpi INTELMPI
```

Using Other appfiles A typical appfile for other machines might look like this:

```
mach1
   mach1
   mach2
   mach2
```

4.2.6. Controlling Files that Distributed ANSYS Writes

Distributed ANSYS writes files to the master and slave machines (or compute nodes) as the analysis progresses. The working directory for each machine can be on a local disk drive or on a network shared drive. For optimal performance, local disk storage is recommended instead of network storage. For more information see the *Performance Guide*. It is strongly recommended that you set up the same working directory path structure on the master and slave machines.

Depending on the system you're running on, you will have varying levels of control over the location of those files. The following explains the expected behavior for each system and each version of MPI, should you wish to control the location of where Distributed ANSYS writes files.

Linux systems with Platform MPI:

- By default, Distributed ANSYS uses the current working directory on the master node and the login ($HOME) directory on the slave nodes.
- You can set the **MPI_WORKDIR** environment variable on the master node to any directory. Distributed ANSYS will then expect that exact directory structure to exist on all of the slave nodes.
- You can set the **MPI_WORKDIR** environment variable on any or all of the slave nodes but NOT the master node. In this scenario, Distributed ANSYS will then use the directory specified by **MPI_WORKDIR** on the nodes where it is set, and will use the current working directory on the master node and the login ($HOME) directory on any slave nodes that do not have **MPI_WORKDIR** set.
- If you are using the launcher, you can select the **Use launcher-specified working directory on all nodes** option on the **High Performance Computing Setup** tab. This option uses the working directory as specified on the **File Management** tab as the directory structure on the master and all nodes. If you select this option, all machines will require the identical directory structure matching the working directory specified on the launcher. This option will override any existing **MPI_WORKDIR** settings on the master or the slave nodes.

- All working directories that are specified must exist before running a job.

Linux systems with Intel MPI:

- By default, you must have identical working directory structures set up on the master and all slave machines. If you don't, Distributed ANSYS will fail to launch.
- Distributed ANSYS will always use the current working directory on the master machine and will expect identical directory structures to exist on all slave nodes. If you are using the launcher, the working directory specified on the **File Management** tab is the directory that Distributed ANSYS will expect.

Windows systems with Microsoft MPI, Platform MPI, Intel MPI:

- By default, you must have identical working directory structures set up on the master and all slave machines. If you don't, Distributed ANSYS will fail to launch.
- Distributed ANSYS will always use the current working directory on the master machine and will expect identical directory structures to exist on all slave nodes. If you are using the launcher, the working directory specified on the **File Management** tab is the directory that Distributed ANSYS will expect.

4.3. Supported Analysis Types and Features

Distributed ANSYS does not currently support all of the analysis types, elements, solution options, and so on that are available in shared-memory ANSYS. In some cases, Distributed ANSYS stops the analysis to avoid performing an unsupported action. If this happens, you must launch shared-memory ANSYS to perform the simulation. In other cases, Distributed ANSYS automatically disables the distributed-memory parallel processing capability and performs the operation using shared-memory parallelism. This disabling of the distributed-memory parallel processing can happen at various levels in the program. The various behavior modes are described in Distributed ANSYS Behavior.

4.3.1. Supported Analysis Types

This section lists analysis capabilities that are supported in Distributed ANSYS and analysis capabilities that are blocked. These are not comprehensive lists, but represent major features and capabilities found in the ANSYS program.

Most element types are valid in an analysis that uses distributed-memory parallel processing (including but not limited to the elements mentioned below). For those element types not supported by Distributed ANSYS, a restriction is included in the element description (see the *Element Reference*). Check the assumptions/restrictions list for the element types you are using.

Supported Analysis Types

The following analysis types are supported and use distributed-memory parallelism throughout the Distributed ANSYS solution. (That is, the solution runs in pure DMP mode.)

- Linear static and nonlinear static analyses for single-field structural problems (DOFs: UX, UY, UZ, ROTX, ROTY, ROTZ, WARP) and single-field thermal problems (DOF: TEMP).
- Buckling analyses using the subspace eigensolver (**BUCOPT**,SUBSP).
- Modal analyses using the Subspace, PCG Lanczos, Unsymmetric, or Damped eigensolver (**MODOPT**,SUBSP; LANPCG; UNSYM; and DAMP).

- Harmonic analyses using the full, Variational Technology, or Variational Technology perfect absorber method (**HROPT**,FULL; VT; VTPA; AUTO).
- Transient dynamic analyses using the full or Variational Technology method (**TRNOPT**,FULL; VT).
- Radiation analyses using the radiosity method.
- Low-frequency electromagnetic analysis using only the following elements: SOLID96, SOLID97, SOLID122, SOLID123, SOLID231, SOLID232, SOLID236, SOLID237, and SOURC36 (when used in a model with the above elements only).
- Coupled-field analyses using only the following elements: PLANE223, SOLID226, SOLID227.
- Superelements in the use pass of a substructuring analysis.
- Cyclic symmetry analyses.

The following analysis types are supported and use distributed-memory parallelism throughout the Distributed ANSYS solution, except for the equation solver which uses shared-memory parallelism. (In these cases, the solution runs in mixed mode.)

- Static and full transient analyses (linear or nonlinear) that use the JCG or ICCG equation solvers. Note that when the JCG equation solver is used in these analysis types, the JCG solver will actually run using distributed-memory parallelism (that is, pure DMP mode) if the matrix is symmetric.
- Buckling analyses using the Block Lanczos eigensolver (**BUCOPT**,LANB).
- Modal analyses using the Block Lanczos or Supernode eigensolver (**MODOPT**,LANB; SNODE).
- Full harmonic analyses using the JCG, ICCG, or QMR equation solvers.

The following analysis types are supported but do not use distributed-memory parallelism within Distributed ANSYS. (The solution runs in pure SMP mode.)

- Modal analyses using the QRDAMP eigensolver (**MODOPT**,QRDAMP).
- Harmonic analyses using the mode superposition method (**HROPT**,MSUP) with symmetric matrices.
- Transient dynamic analyses using the mode superposition method (**TRNOPT**,MSUP).
- Substructure analyses involving the generation pass or expansion pass.
- Spectrum analyses.
- Expansion passes for reduced analyses.

Blocked Analysis Types

- Harmonic analyses using the mode superposition method (**HROPT**,MSUP) with unsymmetric matrices.

4.3.2. Supported Features

This section list features that are supported in Distributed ANSYS and features that are blocked. These are not comprehensive lists, but represent major features and capabilities found in the ANSYS program.

Supported Features:

The following features are supported and use distributed-memory parallelism throughout the Distributed ANSYS solution. (That is, the solution runs in pure DMP mode.)

- Large deformations (**NLGEOM**,ON).
- Line search (**LNSRCH**,ON).
- Auto time stepping (**AUTOTS**,ON).
- Solution controls (**SOLCONTROL**).
- Initial conditions (**IC**).
- Initial state (**INISTATE**).
- Nonlinear material properties specified by the **TB** command.
- Gasket elements and pre-tension elements.
- Lagrange multiplier based mixed u-P elements and TARGE169 - CONTA178.
- Contact nonlinearity (TARGE169 - CONTA178), with the following restriction for the CONTA17*x* elements:

  – KEYOPT(1) = 0, 2 only
  All other KEYOPTS are supported as documented in the contact element descriptions.
- User programmable features, including the user-defined element (USER300).
- Multi-frame restarts.
- Arc-length method (**ARCLEN**).
- Prestress effects (**PSTRES**).
- Mass summary option on the **IRLF** command (**IRLF**,-1).
- Multiple load steps and enforced motion in modal analyses (**MODCONT**).

The following feature is supported, but does not use distributed-memory parallelism within Distributed ANSYS. (The solution runs in pure SMP mode.)

- Residual vectors calculations (**RESVEC**).

Blocked Features:

The following features are not supported by Distributed ANSYS:

- MPC184 rigid link/beam element using the direct elimination method (KEYOPT(2) = 0).
- Variational Technology options on the **STAOPT** and **TRNOPT** commands.
- ANSYS Multi-field solver - multiple code coupling (MFX).
- ANSYS Multi-field solver - single code (MFS).
- Inertia relief (**IRLF**,1).

- Probabilistic design (**/PDS** commands).
- Element morphing.
- Automatic substructuring.

4.4. Understanding the Working Principles and Behavior of Distributed ANSYS

The fundamental difference between Distributed ANSYS and shared-memory ANSYS is that *N* number of ANSYS processes will be running at the same time (where *N* is the total number of CPU processor cores used) for one model. These *N* processes may be running on a single machine (in the same working directory) or on multiple machines. These processes are not aware of each other's existence unless they are communicating (sending messages). Distributed ANSYS, along with the MPI software, provide the means by which the processes communicate with each other in the right location and at the appropriate time.

The following topics give a summary of behavioral differences between Distributed ANSYS and shared-memory ANSYS in specific areas of the ANSYS program:

4.4.1. Differences in General Behavior

File Handling Conventions

Upon startup and during a parallel solution, Distributed ANSYS will append *n* to the current jobname (where *n* stands for the process rank). The master process rank is 0 and the slave processes are numbered from 1 to *N* - 1.

Therefore, upon startup and during a parallel solution, each process will create and use files named `Jobname`*n*`.`*EXT*. These files contain the local data that is specific to each Distributed ANSYS process. Some common examples include the `.LOG` and `.ERR` files as well as most files created during solution such as `.ESAV`, `.FULL`, `.RST`, and `.MODE`. See Files that ANSYS Writes in the *Basic Analysis Guide* for more information on files that ANSYS typically creates.

Other actions (**/PREP7**, **/POST1**, etc.) which are performed only by the master process will work on the global `Jobname`.*EXT* files by default. These files (e.g., `Jobname.DB` and `Jobname.RST`) contain the global data for the entire model. For example, only the master process will save and resume the `Jobname.DB` file. If a non-default file name is specified (e.g., **/ASSIGN**), then Distributed ANSYS behaves the same as shared-memory ANSYS.

After a parallel solution successfully completes, Distributed ANSYS automatically merges some of the (local) `Jobname`*n*`.`*EXT* files into a single (global) file named `Jobname`.*EXT*. These include the `.RST` (or `.RTH`), `.ESAV`, `.EMAT`, `.MODE`, `.FULL`, `.IST`, `.MLV`, and `.SELD` files. This action is performed when the **FINISH** command is executed upon leaving the solution processor. These files contain the same

information about the final computed solution as files generated for the same model computed with shared-memory ANSYS. Therefore, all downstream operations (such as postprocessing) can be performed using shared-memory ANSYS (or in the same manner as shared-memory ANSYS) by using these global files.

If any of these global `Jobname.EXT` files are not needed for downstream operations, you can reduce the overall solution time by suppressing the file combination for individual file types (see the **DMPOPTION** command for more information). If it is later determined that a global `Jobname.EXT` file is needed for a subsequent operation or analysis, the local files can be combined by using the **COMBINE** command.

Distributed ANSYS will not delete most files written by the slave processes when the analysis is completed. If you choose, you can delete these files when your analysis is complete (including any restarts that you may wish to perform). If you do not wish to have the files necessary for a restart saved, you can issue **RESCONTROL**,NORESTART.

File copy, delete, and rename operations can be performed across all processes by using the *DistKey* option on the **/COPY**, **/DELETE**, and **/RENAME** commands. This provides a convenient way to manage local files created by a distributed parallel solution. For example, **/DELETE**,Fname,Ext,,ON automatically appends the process rank number to the specified file name and deletes `Fname`*n*`.Ext` from all processes. See the **/COPY**, **/DELETE**, and **/RENAME** command descriptions for more information.

Batch and Interactive Mode

You can launch Distributed ANSYS in either interactive or batch mode for the master process. However, the slave processes are always in batch mode. The slave processes cannot read the `START150.ANS` or `STOP150.ANS` files. The master process sends all **/CONFIG**,*LABEL* commands to the slave processes as needed.

On Windows systems, there is no ANSYS output console window when running the Distributed ANSYS GUI (interactive mode). All standard output from the master process will be written to a file named `file0.out`. (Note that the jobname is not used.)

Output Files

When a Distributed ANSYS job is executed, the output for the master process is written in the same fashion as shared-memory ANSYS. In other words, by default the output is written to the screen, or if you specified an output file via the launcher or the `-o` command line option, the output for the master process is written to that file. Distributed ANSYS automatically outputs the ASCII files from each slave process to `Jobname`*n*`.OUT`. Normally, these slave process output files have little value because all of the relevant job information is written to the screen (or master process or output file).

Error Handling

The same principle also applies to the error file `Jobname`*n*`.ERR`. When a warning or error occurs on one of the slave processes during the Dis-

tributed ANSYS solution, the process writes that warning or error message to its error file and then communicates the warning or error message to the master process. Typically, this allows the master process to write the warning or error message to its error file and output file and, in the case of an error message, allows all of the Distributed ANSYS processes to exit the program simultaneously.

In some cases, an error message may fail to be fully communicated to the master process. If this happens, you can view each `Jobname`*n*`.ERR` and/or `Jobname`*n*`.OUT` file in an attempt to learn why the job failed. In some rare cases, the job may hang. When this happens, you must manually kill the processes; the error files and output files written by all the processes will be incomplete but may still provide some useful information as to why the job failed.

Use of APDL

In pre- and postprocessing, APDL works the same in Distributed ANSYS as in shared-memory ANSYS. However, in the solution processor (**/SOLU**), Distributed ANSYS does not support certain ***GET** items. In general, Distributed ANSYS supports global solution ***GET** results such as total displacements and reaction forces. It does not support element level results specified by **ESEL**, **ESOL**, and **ETABLE** labels. Unsupported items will return a ***GET** value of zero.

Condensed Data Input

Multiple commands entered in an input file can be condensed into a single line if the commands are separated by the $ character (see Condensed Data Input in the *Command Reference*). Distributed ANSYS cannot properly handle condensed data input. Each command must be placed on its own line in the input file for a Distributed ANSYS run.

4.4.2. Differences in Solution Processing

Domain Decomposition (**DDOPTION** Command)

Upon starting a solution, Distributed ANSYS automatically decomposes the problem into *N* CPU domains so that each process (or each CPU core) works on only a portion of the model. This domain decomposition is done only by the master process and can be controlled using the **DDOPTION** command. The domain decomposition is element-based, which means a CPU domain is a group or subset of elements within the whole model. All FEA data (elements, nodes, materials, sections, real constants, boundary conditions, etc.) required to compute the solution for each CPU domain is communicated to the slave processes by the master process. Throughout the solution, each process works only on its piece of the entire model. When the solution phase ends (e.g., **FINISH** is issued in the solution processor), the master process in Distributed ANSYS works on the entire model again (that is, it behaves like shared-memory ANSYS).

Print Output (**OUTPR** Command)

In Distributed ANSYS, the **OUTPR** command prints NSOL and RSOL in the same manner as in shared-memory ANSYS. However, for other items such as ESOL, Distributed ANSYS prints only the element solution on the CPU domain of the master process. Therefore, **OUTPR**, ESOL has incomplete information and is not recommended. Also, the order of elements is different from that of shared-memory ANSYS due to domain decom-

position. A direct one-to-one element comparison with shared-memory ANSYS will be different if using **OUTPR**.

Large Number of CE/CP and Contact Elements

In all HPC products (both shared-memory ANSYS and Distributed ANSYS), the program can handle a large number of coupling and constraint equations (**CE**/**CP**) and contact elements. However, specifying too many of these items can force Distributed ANSYS to communicate more data among each process, resulting in longer elapsed time to complete a distributed parallel job. You should reduce the number of **CE**/**CP** if possible and make potential contact pairs in a smaller region to achieve non-deteriorated performance. In addition, for assembly contact pairs or small sliding contact pairs, you can use the command **CNCHECK**,TRIM to remove contact and target elements that are initially in far-field (open and not near contact). This trimming option will help to achieve better performance in Distributed ANSYS runs.

4.4.3. Differences in Postprocessing

Postprocessing with Database File and **SET** Commands

Shared-memory ANSYS can postprocess the last set of results using the `Jobname.DB` file (if the solution results were saved), as well as using the `Jobname.RST` file. Distributed ANSYS, however, can only postprocess using the `Jobname.RST` file and cannot use the `Jobname.DB` file as solution results are not entirely written to the database. You will need to issue a **SET** command before postprocessing.

Postprocessing with Multiple Results Files

By default, Distributed ANSYS will automatically combine the local results files (for example, `Jobname`*n*`.RST`) into a single global results file (`Jobname.RST`). This step can be expensive depending on the number of load steps and the amount of results stored for each solution. It requires each local results file to be read by each slave process, communicated to the master process, and then combined together and written by the master process. As a means to reduce the amount of communication and I/O performed by this operation, the **DMPOPTION** command can be used to skip the step of combining the local results file into a single global results file. Then the **RESCOMBINE** command macro can be used in **/POST1** to individually read each local results file until the entire set of results is placed into the database for postprocessing. If needed, a subsequent **RESWRITE** command can then be issued to write a global results file for the distributed solution.

Note that if the step of combining the results file is skipped, it may impact downstream analyses that rely on a single global results file for the entire model. If it is later determined that a global results file (e.g., `Jobname.RST`) is needed for a subsequent operation, you can use the **COMBINE** command to combine the local results files into a single, global results file.

4.4.4. Restarts in Distributed ANSYS

Distributed ANSYS supports multiframe restarts for nonlinear static and full transient analyses. The procedures and command controls are the same as described in the *Basic Analysis Guide*; however, restarts in Distributed ANSYS have additional limitations:

- You cannot perform a restart using shared-memory parallel processing (shared-memory ANSYS) if Distributed ANSYS was used prior to the restart point.
- The total number of cores used when restarting Distributed ANSYS must not be altered following the first load step and first substep.

  For example, if you use the following command line for the first load step:

  ```
  ansys150 -dis -np 8 -i input -o output1
  ```

  then, for the multiframe restart you must also use 8 cores (`-dis -np 8`), and the files from the original analysis that are required for the restart must be located in the current working directory.
- When running across machines, the job launch procedure (or script) used when restarting Distributed ANSYS must not be altered following the first load step and first substep. In other words, you must use the same number of machines, the same number of cores for each of the machines, and the same host (master) and slave relationships among these machines in the restart job that follows.

  For example, if you use the following command line for the first load step:

  ```
  ansys150 –dis –machines mach1:4:mach2:1:mach3:2 –i input –o output1
  ```

  where the host machine (which always appears first in the list of machines) is `mach1`, and the slave machines are `mach2` and `mach3`. Then for the multiframe restart, you must use a command line such as this:

  ```
  ansys150 –dis –machines mach7:4:mach6:1:mach5:2 –i restartjob –o output2
  ```

  which uses the same number of machines (3), same number of cores for each machine in the list (4:1:2), and the same host/slave relationship (4 cores on host, 1 core on first slave, and 2 cores on second slave). Any alterations in the `-machines` field, other than the actual machine names, will result in restart failure. Finally, the files from the original analysis that are required for the restart must be located in the current working directory on each of the machines.
- The files needed for a restart must be available on the machine(s) used for the restarted analysis. Each machine has its own restart files that are written from the previous run. The restart process needs to use these files to perform the correct restart actions.

  For the first example above, if the two analyses (`-dis –np 8`) are performed in the same working directory, no action is required; the restart files will already be available. However, if the restart is performed in a new directory, all of the restart files listed in Table 4.4: Required Files for Multiframe Restarts (p. 39) must be copied (or moved) into the new directory before performing the multiframe restart.

  For the second example above, the restart files listed in the "Host Machine" column in Table 4.4: Required Files for Multiframe Restarts (p. 39) must be copied (or moved) from `mach1` to `mach7`, and all of the files in the "Slave Machines" column must be copied (or moved) from `mach2` to `mach6` and from `mach3` to `mach5` before performing the multiframe restart.

Table 4.4: Required Files for Multiframe Restarts

Host Machine	Slave Machines
`Jobname.LDHI`	- -
`Jobname.RDB`	- -
`Jobname0.`*`Xnnn`* [1]	`Jobname`*`n`*`.`*`Xnnn`*

Host Machine	Slave Machines
Jobname0.RST (this is the local .RST file for this domain)	Jobname*n*.RST (this is the local .RST file for this domain)

1. The .X*nnn* file extension mentioned here refers to the .R*nnn* and .M*nnn* files discussed in Multiframe File Restart Requirements in the *Basic Analysis Guide*.

In all restarts, the result file Jobname.RST (or Jobname.RTH or Jobname.RMG) on the host machine is recreated after each solution by merging the Jobname*n*.RST again.

If you do not require a restart, issue **RESCONTROL**,NORESTART in the run to remove or to avoid writing the necessary restart files on the host and slave machines. If you use this command, the slave processes will not have files such as .ESAV, .OSAV, .RST, or .X000, in the working directory at the end of the run. In addition, the host process will not have files such as .ESAV, .OSAV, .X000, .RDB, or .LDHI at the end of the run. The program will remove all of the above scratch files at the end of the solution phase (**FINISH** or **/EXIT**). This option is useful for file cleanup and control.

4.5. Example Problems

This section contains tutorials for running Distributed ANSYS on Linux and Windows platforms.

This tutorial is divided into two parts.

The first part walks you through setting up your distributed environment and running a test to verify that communication between systems is occurring correctly. Running this part is optional (though strongly recommended), and need only be done when you initially configure your environment, or when you change your environment by adding, removing, or changing machines.

The second part of the tutorial walks you through setting up your distributed environment and then directly running a sample problem. The tutorial is designed so that you can modify settings (such as the number of machines in the cluster, number of cores on each machine, etc.), but we strongly recommend that the first time you run this tutorial, you follow the steps exactly. Once you're familiar with the process, you can then modify the tutorial to more closely match your particular environment. You can then use the files generated in future GUI or batch runs.

Sample inputs to use for each example problem are included on the media under Program Files\Ansys Inc\V150\ansys\Data\models (Windows) or /ansys_inc/v150/ansys/data/models (Linux). Static and modal examples are provided. The problem setup for each platform is the same whether you run the static or the modal example.

4.5.1. Example: Running Distributed ANSYS on Linux

The following tutorial walks you through the setup of your Distributed ANSYS environment, and is applicable only to systems running ANSYS 15.0 on a 64-bit Linux cluster under Platform MPI 9.1. The ANSYS 15.0 installation includes Platform MPI 9.1.

One of the sample problems, tutor1_carrier_linux.inp (static) or tutor2_carrier_modal.inp (modal), is required to complete the tutorial. Save this file to your working directory before beginning the tutorial. You can run either sample problem using the problem setup described here.

Part A: Setup and Run mpitest

1. Set up identical installation and working directory structures on all machines (master and slaves) in the cluster.

2. Type **hostname** on each machine in the cluster. Note the name of each machine. You will need this name to set up both the `.rhosts` file and the **Configure Cluster** option of the **ANS_ADMIN150** utility.

   Set up the `.rhosts` file on each machine. The `.rhosts` file lists each machine in the cluster, followed by your username. The machines should be listed using their complete system name, as taken from **uname**. For example, each `.rhosts` file for our two-machine cluster looks like this (where golinux1 and golinux2 are example machine names, and jqd is an example username):

   ```
   golinux1 jqd
   golinux2 jqd
   ```

   Change/verify `.rhosts` file permissions on all machines by issuing:

   ```
   chmod 600 .rhosts
   ```

   Navigate to your working directory. Run the following:

   ```
   /ansys_inc/v150/ansys/bin/mpitest150
   ```

The mpitest program should start without errors. If it does not, check your paths, .rhosts file, and permissions; correct any errors; and rerun.

Part B: Setup and Run a Distributed Solution

1. Set up identical installation and working directory structures on all machines (master and slaves) in the cluster.

2. Install ANSYS 15.0 on the master machine, following the typical installation process.

3. Install ANSYS 15.0 on the slave machines.

   Steps 2 and 3 above will install all necessary components on your machines, including Platform MPI 9.1.

4. Type **hostname** on each machine in the cluster. Note the name of each machine. You will need this name to set up both the `.rhosts` file and the **Configure Cluster** option of the **ANS_ADMIN150** utility.

5. Set up the `.rhosts` file on each machine. The `.rhosts` file lists each machine in the cluster, followed by your username. The machines should be listed using their complete system name, as taken from **uname**. For example, each `.rhosts` file for our two-machine cluster looks like this (where golinux1 and golinux2 are example machine names, and jqd is an example username):

   ```
   golinux1 jqd
   golinux2 jqd
   ```

6. Change/verify `.rhosts` file permissions on all machines by issuing:

   ```
   chmod 600 .rhosts
   ```

7. Verify communication between machines via rsh. If the communication between machines is happening correctly, you will not need a password.

8. Run the **ANS_ADMIN150** utility:

```
/ansys_inc/v150/ansys/bin/ans_admin150
```

9. Choose **ANSYS / Workbench Configuration Cluster**. Under **Select file to configure**, choose the `hosts150.ans` file to be configured and choose **Configure for Distributed ANSYS**. Click **OK**. Then enter the system name (from Step 4) in the **Machine hostname** field and click **Add**.

   On the next dialog box, enter the system type in the **Machine type** drop-down, and the number of cores in the **Max number of jobs/processors** field for each machine in the cluster. Click **OK**. When you are finished adding machines, click **Close** and then click **Exit** to leave the **ANS_ADMIN150** utility. The resulting `hosts150.ans` file using our example machines would look like this:

```
golinux1 linem64t 0 1 0 0 /home/jqd MPI 1 1
golinux2 linem64t 0 1 0 0 /home/jqd MPI 1 1
```

10. Start ANSYS using the launcher:

```
launcher150
```

11. Select the correct environment and license.

12. Go to the **High Performance Computing Setup** tab. Select **Use Distributed Computing (MPP)**. You must also specify either local machine or multiple hosts. For multiple hosts, select the machines you want to use from the list of available hosts. The list of available hosts is populated from the `hosts150.ans` file. Click on the machines you want to use and click **Add** to move them to the Selected Hosts list to use them for this run. You can also add or remove a host, but be aware that adding or removing a host from here will modify only this run; the `hosts150.ans` file will not be updated with any new information from this dialog box.

    If necessary, you can also run secure shell (ssh) by selecting **Use Secure Shell instead of Remote Shell (ssh instead of rsh)**.

13. Click **Run** to launch ANSYS.

14. In ANSYS, select **File>Read Input From** and navigate to `tutor1_carrier_linux.inp` or `tutor2_carrier_modal.inp`.

15. The example will progress through the building, loading, and meshing of the model. When it stops, select **Main Menu>Solution>Analysis Type>Sol'n Controls**.

16. On the **Solution Controls** dialog box, click on the **Sol'n Options** tab.

17. Select the **Pre-Condition CG solver**.

18. Click **OK** on the **Solution Controls** dialog box.

19. Solve the analysis. Choose **Main Menu>Solution>Solve>Current LS**. Click **OK**.

20. When the solution is complete, you can postprocess your results as you would with any analysis. For example, you could select **Main Menu>General Postproc>Read Results>First Set** and select the desired result item to display.

4.5.2. Example: Running Distributed ANSYS on Windows

The following tutorial walks you through the setup of your Distributed ANSYS environment, and is applicable only to systems running ANSYS 15.0 on a Windows cluster under Platform MPI 9.1.

One of the sample problems, `tutor1_carrier_win.inp` (static) or `tutor2_carrier_modal.inp` (modal), is required to complete the tutorial. These files are found in `Program Files\Ansys Inc\V150\ansys\Data\models`. Copy the file you want to use to your working directory before beginning the tutorial. You can run either sample problem using the problem setup described here.

1. Set up identical installation and working directory structures on all machines (master and slaves) in the cluster.

2. Install ANSYS 15.0 on the master machine, following the typical installation process.

3. Configure ANSYS 15.0 on the slave machines.

4. Install and register Platform MPI 9.1 on both machines following the instructions in Prerequisites for Running Distributed ANSYS (p. 19).

5. Add `%MPI_ROOT%\bin` to the **PATH** environmental variable on both machines (assuming Platform MPI 9.1 was installed on the `C:\` drive). This line must be in your path for the mpirun command to be recognized.

6. On each machine, right-click on **My Computer**, left-click on **Properties**, and select the **Network Identification** or **Computer Name** tab. The full computer name will be listed. Note the name of each machine (not including the domain). You will need this name to set up the **Configure Cluster** option of the **ANS_ADMIN** utility.

7. Run the **ANS_ADMIN** utility on the master machine: **Start >Programs >ANSYS 15.0 >Utilities >ANS_ADMIN 15.0**.

8. Choose **ANSYS / Workbench Configure Cluster** to configure the `hosts150.ans` file.

   - Specify the directory in which the `hosts150.ans` will be configured (two possible locations are presented). Also select **Configure for ... Distributed ANSYS**. Click **OK**.

   - Enter the system name (from Step 6) in the **Machine hostname** field and click **Add**. On the next dialog box, enter the system type in the **Machine type** drop-down, and the number of cores in the **Max number of jobs/processors** field and click **OK**. Repeat this step for each machine in the cluster. When you are finished adding machines, click **Close** and then **File >Exit**. The resulting `hosts150.ans` file using our example machines where machine1 has 2 cores and machine2 has 4 cores would look like this:

     ```
     machine1 intel 0 2 0 0 MPI 1 1
     machine2 intel 0 4 0 0 MPI 1 1
     ```

9. Start ANSYS using the launcher: **Start >Programs >ANSYS 15.0 > Mechanical APDL Product Launcher 15.0**.

10. Select ANSYS Batch as the Simulation Environment, and choose a license. Specify `tutor1_carrier_win.inp` or `tutor2_carrier_modal.inp` as your input file. Both of these examples use the PCG solver. You must specify your working directory to be the location where this file is located.

11. Go to the **High Performance Computing Setup** tab. Select **Use Distributed Computing (MPP)**. You must specify either local machine or multiple hosts. For multiple hosts, select the machines you want to use from the list of available hosts. The list of available hosts is populated from the `hosts150.ans` file. Click on the machines you want to use and click **Add** to move them to the Selected Hosts list to use them for this run. Click on a machine in Selected Hosts and click **Edit** if you wish to add multiple cores for that host. You can also add or remove a host, but be aware that adding or removing a host from here will modify only this run; the `hosts150.ans` file will not be updated with any new information from this dialog box.

12. Click **Run**.

13. When the solution is complete, you can postprocess your results as you would with any analysis.

4.6. Troubleshooting

This section describes problems which you may encounter while using Distributed ANSYS, as well as methods for overcoming these problems. Some of these problems are specific to a particular system, as noted.

4.6.1. Setup and Launch Issues

To aid in troubleshooting, you may need to view the actual MPI run command line. On Linux the command is mpirun, and you can view the command line by setting the **ANS_SEE_RUN_COMMAND** environment variable to 1. On Windows the command is mpiexec, and you can view the command line by setting the **ANS_SEE_RUN** and **ANS_CMD_NODIAG** environment variables to TRUE.

Job fails to launch

The first thing to check when a Distributed ANSYS job fails to launch is that the MPI software you wish to use is installed and properly configured (see Configuring Distributed ANSYS (p. 19)).

Next, if running across multiple machines, ensure that the working directory path is identical on all machines (or that you are using a shared network directory) and that you have permission to write files into the working directory used by each machine.

Finally, make sure that you are running the distributed solution on a homogeneous cluster. The OS level and processors must be identical on all nodes in the cluster. If they are not, you may encounter problems. For example, when running Distributed ANSYS across machines using Intel MPI, if the involved cluster nodes have different processor models, the program may hang (i.e., fail to launch). In this situation, no data is written to any files and no error message is output.

No permission to system

This error can occur if you do not have login access to a remote system where Distributed ANSYS is supposed to run. If you use Linux, you can experience this problem if you have not set permissions on the `.rhosts` file properly. Before starting Distributed ANSYS, be sure you have access to all the remote systems you have specified (i.e., you should be able to rsh to those systems) and that the control node is included in its own `.rhosts` file. If you run on Linux, be sure you have run the following command:

```
chmod 600 .rhosts
```

MPI: could not run executable

If you encounter this message, verify that you have the correct version of MPI and that it is installed correctly, and verify that you have a `.rhosts` file on each machine. If not, create a `.rhosts` file on all machines where you will run Distributed ANSYS, make sure the permissions on the file are 600, and include an entry for each hostname where you will run Distributed ANSYS.

Error executing ANSYS. Refer to System-related Error Messages in the ANSYS online help. If this was a Distributed ANSYS job, verify that your MPI software is installed correctly, check your environment settings, or check for an invalid command line option.

You may encounter the above message when setting up Platform MPI or running Distributed ANSYS using Platform MPI on a Windows platform. This may occur if you did not correctly run the set password bat file. Verify that you completed this item according to the Platform MPI installation readme instructions.

You may also see this error if `Ansys Inc\v150\ansys\bin\<platform>` (where <*platform*> is intel or winx64) is not in your PATH.

If you need more detailed debugging information, use the following:

1. Open a Command Prompt window and set the following:

```
SET ANS_SEE_RUN=TRUE
SET ANS_CMD_NODIAG=TRUE
```

2. Run the following command line: **`ansys150 -b -dis -i myinput.inp -o myoutput.out`**.

Distributed ANSYS fails to launch when running from a fully-qualified pathname.

Distributed ANSYS will fail if the ANSYS 15.0 installation path contains a **space followed by a dash** if **`%ANSYS150_DIR%\bin\<platform>`** (where <*platform*> is intel or winx64) is not in the system PATH. Add **`%ANSYS150_DIR%\bin\<platform>`** to the system PATH and invoke **`ansys150`** (without the fully qualified pathname). For example, if your installation path is:

```
C:\Program Files\Ansys -Inc\v150\bin\<platform>
```

The following command to launch Distributed ANSYS will fail:

```
"C:\Program Files\Ansys -Inc\v150\bin\<platform>\ansys150.exe" -g
```

However, if you add **`C:\Program Files\Ansys -Inc\v150\bin\<platform>`** to the system PATH, you can successfully launch Distributed ANSYS by using the following command:

```
ansys150 -g
```

The required licmsgs.dat file, which contains licensing-related messages, was not found or could not be opened. The following path was determined using environment variable ANSYS150_DIR. This is a fatal error - - exiting.

Check the **ANSYS150_DIR** environment variable to make sure it is set properly. Note that for Windows HPC clusters, the **ANSYS150_DIR** environment variable should be set to \\HEADNODE\Ansys Inc\v150\ansys, and the **ANSYSLIC_DIR** environment variable should be set to \\HEADNODE\Ansys Inc\Shared Files\Licensing on all nodes.

mpid: Cannot set work directory: No such file or directory mpid: MPI_WORKDIR=<*dir*>

You will see this message if you set the **MPI_WORKDIR** environment variable but the specified directory doesn't exist. If you set **MPI_WORKDIR** on the master, this message could also appear if

the directory doesn't exist on one of the slaves. In Distributed ANSYS, if you set the **MPI_WORKDIR** environment variable on the master node, Distributed ANSYS will expect all slave nodes to have the same directory.

WARNING: Unable to read mpd.hosts or list of hosts isn't provided. MPI job will be run on the current machine only. (Intel MPI)

When using Intel MPI, you must have an mpd.hosts file in your working directory when going across boxes that contain a line for each box. Otherwise, you will encounter the following error.

WARNING: Unable to read mpd.hosts or list of hosts isn't provided. MPI job will be run on the current machine only.
mpiexec: unable to start all procs; may have invalid machine names
remaining specified hosts:
xx.x.xx.xx (hostname)

mpid: MPI BUG: requested interconnect not available

The default locations for GM (Myrinet) and its libraries are `/opt/gm` and `/opt/gm/lib`, respectively. If the libraries are not found, you may see this message. To specify a different location for the GM libraries, set the **MPI_ICLIB_GM** environment variable:

```
setenv MPI_ICLIB_GM <path>/lib/libgm.so
```

AMD Opteron and other 64-bit systems may have a specific 64-bit library subdirectory, `/lib64`. On these systems, you need to point to this location:

```
setenv MPI_ICLIB_GM <path>/lib64/libgm.so
```

Note

The environment variable needs to be set on each system (such as in the `.cshrc` or `.login` files).

4.6.2. Solution and Performance Issues

This section describes solution and performance issues that you may encounter while running Distributed ANSYS.

Recovering from a Computer, Network, or Program Crash

When a Distributed ANSYS job crashes unexpectedly (e.g., seg vi, floating point exception, out-of-disk space error), an error message may fail to be fully communicated to the master process and written into the output file. If this happens, you can view all of the output and/or error files written by each of the slave processes (e.g., `Jobname`*n*`.OUT` and/or `Jobname`*n*`.ERR`) in an attempt to learn why the job failed. In some rare cases, the job may hang. When this happens, you must manually kill the processes; the error files and output files written by all the processes will be incomplete but may still provide some useful information as to why the job failed.

Be sure to kill any lingering processes (Linux: type **kill -9** from command level; Windows: use **Task Manager**) on all processors and start the job again.

Job fails with SIGTERM signal (Linux Only)

Occasionally, when running on Linux, a simulation may fail with a message like the following:

MPI Application rank 2 killed before MPI_Finalize() with signal 15

forrtl: error (78): process killed (SIGTERM)

This typically occurs when computing the solution and means that the system has killed the ANSYS process. The two most common occurrences are (1) ANSYS is using too much of the hardware resources and the system has killed the ANSYS process or (2) a user has manually killed the ANSYS job (i.e., **kill -9** system command). Users should check the size of job they are running in relation to the amount of physical memory on the machine. Most often, decreasing the model size or finding a machine with more RAM will result in a successful run.

Poor Speedup or No Speedup

As more cores are utilized, the runtimes are generally expected to decrease. The biggest relative gains are typically achieved when using two cores compared to using a single core. When significant speedups are not seen as additional cores are used, the reasons may involve both hardware and software issues. These include, but are not limited to, the following situations.

Hardware

Oversubscribing hardware In a multiuser environment, this could mean that more physical cores are being used by multiple simulations than are available on the machine. It could also mean that hyperthreading is activated. Hyperthreading typically involves enabling extra virtual cores, which can sometimes allow software programs to more effectively use the full processing power of the CPU. However, for compute-intensive programs such as ANSYS, using these virtual cores rarely provides a significant reduction in runtime. Therefore, it is recommended you do not use hyperthreading; if hyperthreading is enabled, it is recommended you do not exceed the number of physical cores.

Lack of memory bandwidth On some systems, using most or all of the available cores can result in a lack of memory bandwidth. This lack of memory bandwidth can impact the overall scalability.

Slow interconnect speed When running Distributed ANSYS across multiple machines, the speed of the interconnect (GigE, Myrinet, Infiniband, etc.) can have a significant impact on the performance. Slower interconnects cause each Distributed ANSYS process to spend extra time waiting for data to be transferred from one machine to another. This becomes especially important as more machines are involved in the simulation. See Interconnect Configuration for the recommended interconnect speed.

Software

Simulation includes non-supported features The shared and distributed-memory parallelisms work to speed up certain compute-intensive operations in **/PREP7**, **/SOLU** and **/POST1**. However, not all operations are parallelized. If a particular operation that is not parallelized dominates the simulation time, then using additional cores will not help achieve a faster runtime.

Simulation has too few DOF (degrees of freedom) Some analyses (such as transient analyses) may require long compute times, not because the number of DOF is large, but because a large number of calculations are performed (i.e., a very large number of time steps). Generally, if the number of DOF is relatively small, parallel processing will not significantly decrease the solution time. Consequently, for small models with many time steps, parallel performance may be poor because the model size is too small to fully utilize a large number of cores.

I/O cost dominates solution time For some simulations, the amount of memory required to obtain a solution is greater than the physical memory (i.e., RAM) available on the machine. In these cases, either virtual memory (i.e., hard disk space) is used by the operating system to hold the data that would otherwise be stored in memory, or the equation solver writes extra files to the disk to store data. In both cases, the extra I/O done using the hard drive can significantly

impact performance, making the I/O performance the main bottleneck to achieving optimal performance. In these cases, using additional cores will typically not result in a significant reduction in overall time to solution.

Large contact pairs For simulations involving contact pairs with a large number of elements relative to the total number of elements in the entire model, the performance of Distributed ANSYS is often negatively impacted. These large contact pairs require Distributed ANSYS to do extra communication and often cause a load imbalance between each of the cores (i.e., one core might have two times more computations to perform than another core). In some cases, using **CNCHECK**,TRIM can help trim any unnecessary contact/target elements from the larger contact pairs. In other cases, however, manual interaction will be required to reduce the number of elements involved in the larger contact pairs.

Different Results Relative to a Single Core

Distributed-memory parallel processing initially decomposes the model into domains. Typically, the number of domains matches the number of cores. Operational randomness and numerical round-off inherent to parallelism can cause slightly different results between runs on the same machine(s) using the same number of cores or different numbers of cores. This difference is often negligible. However, in some cases the difference is appreciable. This sort of behavior is most commonly seen on nonlinear static or transient analyses which are numerically unstable. The more numerically unstable the model is, the more likely the convergence pattern or final results will differ as the number of cores used in the simulation is changed.

Index