Intel® MPI Library for Windows* OS

User's Guide

Document Number: 316404-011

Contents

Legal Information

1. Introduction

The *Intel® MPI Library for Windows* OS User Guide* explains how to use the Intel® MPI Library in some common usage scenarios. It provides information regarding compiling, linking, running, and debugging MPI applications, as well as information on integration within a cluster environment.

This *User's Guide* contains the following sections

Document Organization

Section	**Description**
Section 1 Introduction	Section 1 introduces this document
Section 2 Usage Model	Section 2 presents the usage model for working with the Intel® MPI Library
Section 3 Installation and Licensing	Section 3 describes the installation process and provides information about licensing
Section 4 Compiling and Linking	Section 4 gives instructions about how to compile and link MPI applications
Section 5 Running Applications	Section 5 describes the steps for running an application
Section 6 Debugging and Testing	Section 6 explains how to start an application under a debugger
Section 7 Process Management	Section 7 gives information about process managers
Section 8 Tuning with mpitune Utility	Section 8 describes how to use the mpitune utility to find optimal settings for the Intel® MPI Library.
Section 9 Job Schedulers Support	Section 9 describes integration with job schedulers
Section 10 General Cluster Considerations	Section 10 discusses general considerations for clusters related to MPI usage

1.1. Introducing Intel® MPI Library

The Intel® MPI Library is a multi-fabric message passing library that implements the Message Passing Interface, version 3.0 (MPI-3.0) specification. It provides a standard library across Intel® platforms that:

- Delivers best in class performance for enterprise, divisional, departmental and workgroup high performance computing. The Intel® MPI Library focuses on improving application performance on Intel® architecture based clusters.
- Enables you to adopt MPI-3.0 functions as your needs dictate

1.2. Intended Audience

This *User's Guide* helps an experienced user to start using the Intel® MPI Library and contains brief descriptions of the main functionality as a set of how-to instructions and examples. For full information, see *Intel® MPI Library Reference Manual for Windows* OS*.

1.3. Conventions and Symbols

The following conventions are used in this document.

Table 1.3-1 Conventions and Symbols used in this Document

This type style	Document or product names
This type style	Hyperlinks
This type style	Commands, arguments, options, file names
`THIS_TYPE_STYLE`	Environment variables
`<this type style>`	Placeholders for actual values
`[ items ]`	Optional items
`{ item | item }`	Selectable items separated by vertical bar(s)
(SDK only)	For Software Development Kit (SDK) users only

1.4. Related Information

To get more information about the Intel® MPI Library, explore the following resources:

- *Intel® MPI Library Release Notes* for updated information on requirements, technical support, and known limitations.

- *Intel® MPI Library Reference Manual* for in-depth knowledge of the product features, commands, options, and environment variables.

- *Intel® MPI Library for Windows* OS Knowledge Base* for additional troubleshooting tips and tricks, compatibility notes, known issues, and technical notes.

For additional resources, see:

Intel® MPI Library Product Web Site

Intel Product Support

Intel® Cluster Tools Products Website

Intel® Software Development Products Website

2. Usage Model

Using the Intel® MPI Library involves the following steps:

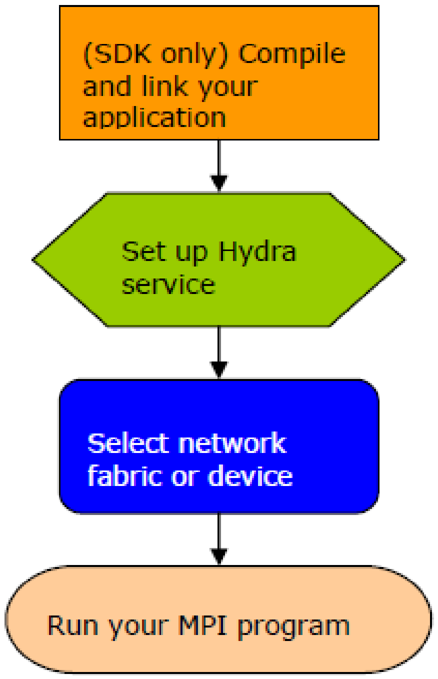
(SDK only) Compile and link your application
Set up Hydra service
Select network fabric or device
Run your MPI program

Figure 1: Flowchart representing the usage model for working with the Intel® MPI Library.

3. Installation and Licensing

This section describes the installation process and provides information about licensing for the following products:

- Intel® MPI Library
- Intel® MPI Library Runtime Environment (RTO)
- Intel® MPI Library Software Development Kit (SDK)

3.1. Installing Intel® MPI Library

If you have a previous version of the Intel® MPI Library for Windows* OS installed, you do not need to uninstall it before installing the latest version.

To install the Intel MPI Library, double-click on the distribution file `w_mpi_p_<version>.<package_num>.exe` (SDK), and `w_mpi-rt_p_<version>.<package_num>.exe` (RTO).

You will be asked to choose a directory in which the contents of the self-extracting installation file will be placed before the actual installation begins. After installation, the files will still be located in this directory. By default, the `C:\Program Files (x86)\Intel\Download` is used on machines with Intel® 64 architecture.

Follow the prompts outlined by the installation wizard to complete the installation. See full description of the installation process and requirements in the *Intel® MPI Library for Windows* OS Installation Guide*. You can also find information about silent installation and cluster installation in the installation guide.

NOTE

You need the domain administrator right when you install the Intel® MPI Library on Windows* OS. If you do not have domain administrator permissions, you cannot proceed with the Active Directory* setup on Windows* OS. See the *Intel® MPI Library Reference Manual* for more Active Directory* setup information.

3.2. Licensing for Intel® MPI Library Runtime Environment (RTO) and Intel® MPI Library Software Development Kit (SDK)

There are two different licensing options:

- Intel® MPI Library Runtime Environment (RTO) license. This license includes tools you need to run MPI programs including runtime process managers, supporting utilities, shared (.dll) libraries and documentation. The license includes everything you need to run Intel® MPI Library-based applications and is free and permanent.

- Intel® MPI Library Software Development Kit (SDK) license. This license includes all of Runtime Environment components as well as the compilation tools: compiler commands (`mpiicc`, `mpicc`, and so on.), files and modules, static (.lib) libraries, debug libraries, trace libraries, and test sources. This license is fee-based, with the following options:

    - evaluation
    - named-user
    - floating

For licensing details refer to the EULA or visit http://www.intel.com/go/mpi.

4. Compiling and Linking

This section gives instructions about how to compile and link different kinds of your Intel® MPI Library applications, and details on different debugging and compiler support options.

4.1. Compiling a Serial Program

(SDK only)

To compile and link an MPI program with the Intel® MPI Library do the following steps:

1. In Microsoft* Visual Studio*, create a Winxx Console project
2. Choose the x64 solution platform.
3. Add `<installdir>\intel64\include` to the include path.
4. Add `<installdir>\intel64\lib\<configuration>` to the library path. You can set the following values for `<configuration>`:
    - `Debug`: single-threaded debug library.
    - `Release`: single-threaded optimized library.
    - `Debug_mt`: multi-threaded debug library.
    - `Release_mt`: multi-threaded optimized library.
5. Add the appropriate Intel MPI libraries to your target links command:
    - Add `impi.lib` for C applications.
    - Add `impi.lib` and `impicxx.lib` (Release) or `impi.lib` and `impicxxd.lib` (Debug) for C++ applications.
6. Compile and build your program.
7. Place your application and all the dynamic libraries in a shared location or copy them to all the nodes.
8. Run the application using the `mpiexec.exe` command.

4.2. Adding Debug Symbols

If you need to debug your application, add the `-g` option. In this case debug information is added to the binary. You can use any debugger to debug the application.

```
> mpiicc test.c -o testc -g
```

4.3. Other Compilers Support

In addition to the latest Intel® compilers, the Intel® MPI Library offers support for the Microsoft* Visual Studio* compilers. Check the *Intel® MPI Library Release Notes for Windows* OS* for a full list of supported Microsoft Visual Studio versions. Check the Microsoft Visual Studio documentation for details on how to link with third party libraries.

5. Running Applications

After you have compiled and linked your application, you are ready to run your Intel MPI applications. This topic describes the steps for running applications.

5.1. Running an Intel® MPI Program

To launch programs linked with the Intel® MPI Library use the `mpiexec` command:

```
> mpiexec.exe -n <# of processes> myprog.exe
```

The `wmpiexec` utility is a GUI wrapper for `mpiexec.exe`. See the Intel® MPI Library Reference Manual for more details.

To set the number of processes on the local node, use the `mpiexec -n` option.

To set names of hosts and number of processes, use the `-hosts` option:

```
> mpiexec.exe -hosts 2 host1 2 host2 2 myprog.exe
```

If you are using a network fabric as opposed to the default fabric, use the `-genv` option to set the `I_MPI_FABRICS` variable.

For example, to run an MPI program using the shm fabric, type in the following command:

```
> mpiexec.exe -genv I_MPI_FABRICS shm -n <# of processes> myprog.exe
```

You may use the `-configfile` option to define the configuration options in a separate file and simplify your execution command:

```
> mpiexec.exe -configfile config_file
```

In this example, the configuration file contains:

```
-genv I_MPI_FABRICS shm:dapl

-host host1 -n 1 myprog.exe

-host host2 -n 1 myprog.exe
```

This allows you to run two versions of our executable (`myprog.exe`) on separate hosts, both using the faster dapl fabric.

For RDMA-capable networks, use the following command:

```
> mpiexec.exe -hosts 2 host1 1 host2 1 -genv I_MPI_FABRICS dapl
myprog.exe
```

For more information, see Selecting Fabrics.

After you successfully run your application using the Intel® MPI Library, you can move your application from one cluster to another and use different fabrics between the nodes without re-linking. If you encounter problems, see Debugging and Testing for possible solutions.

5.2. Multi-threaded Applications

To run an application which creates new threads, you need to link your application with thread safe version of the Intel® MPI Library. You do not need any extra option while code compilation, the compiler script like `mpiicc` links code with thread safe library by default.

To run OpenMP* application and pin threads inside the domain, make sure the `KMP_AFFINITY` environment variable is set to use the corresponding OpenMP* feature.

Run the application:

```
> mpiexec.exe -genv OMP_NUM_THREADS 4 -n <# of processes> .\myprog
```

For more details see the Interoperability with OpenMP* topic in the *Intel® MPI Library Reference Manual for Windows* OS*.

To switch to multi-threaded debug library configuration, use the following command:

```
> call mpivars.bat debut_mt
```

5.3. Selecting Fabrics

By default, the Intel® MPI Library selects a network fabric based on the list of fabrics specified in `I_MPI_FABRICS_LIST`. To select a specific network fabric combination, use the `-genv` option to assign a value to the `I_MPI_FABRICS` variable. You can also assign a value using the export command.

If the specified fabric is not available, Intel® MPI Library will go down the list specified in `I_MPI_FABRICS_LIST` and select the next available fabric.

You can disable this fallback behavior by using the `I_MPI_FALLBACK` variable:

```
-genv I_MPI_FALLBACK 0
```

By default, the fallback is enabled. If `I_MPI_FABRICS` is set, the fallback will be disabled.

TCP Socket Connection

Use the following command to run an MPI program over TCP sockets using the available Ethernet connection on the cluster. The program does not use shared memory within a node:

```
    > mpiexec.exe -genv I_MPI_FABRICS tcp -n <# of processes>
./myprog.exe
```

Shared Memory

Use the following command to run an MPI program over the shared-memory fabric (shm) only:

```
    > mpiexec.exe -genv I_MPI_FABRICS shm -n <# of processes>
./myprog.exe
```

Shared Memory and DAPL* Connection

To use shared memory for intra-node communication and the Direct Access Programming Library* (DAPL*) layer for inter-node communication, use the following command:

```
> mpiexec.exe -genv I_MPI_FABRICS shm:dapl -n <# of processes>
./myprog.exe
```

This is the default method if no fabric options are selected.

For more details see the Fabrics Control topic in *Intel® MPI Library Reference Manual for Windows* OS*.

After you successfully run your application using the Intel MPI Library over any of the fabrics described, you can move your application from one cluster to another and use different fabrics between the nodes without re-linking. If you encounter problems, see Debugging and Testing for possible solutions.

5.3.1. I_MPI_FABRICS

This topic is an excerpt from the *Intel® MPI Library Reference Manual for Windows* OS* which provides further details on the `I_MPI_FABRICS` environment variable.

Select a particular network fabric to be used for communication.

Syntax

```
I_MPI_FABRICS=<fabric>|<intra-node fabric>:<inter-nodes fabric>
```

Where

- `<fabric> := {shm, dapl, tcp}`
- `<intra-node fabric> := {shm, dapl, tcp}`
- `<inter-nodes fabric> := {dapl,tcp}`

Arguments

Argument	Definition
<fabric>	Define a network fabric
shm	Shared-memory
dapl	DAPL-capable network fabrics, such as InfiniBand*, iWarp*, Dolphin*, and XPMEM* (through DAPL*)
tcp	TCP/IP-capable network fabrics, such as Ethernet and InfiniBand* (through IPoIB*)

For example, to select the winOFED* InfiniBand* device, use the following command:

```
>mpiexec.exe -n <# of processes>  \

-env I_MPI_FABRICS shm:dapl <executable>
```

For these devices, if *<provider>* is not specified, the first DAPL* provider in the `dat.conf` file is used. The `shm` fabric is available for both Intel® and non-Intel microprocessors, but it may perform additional optimizations for Intel microprocessors than for non-Intel microprocessors.

NOTE

Ensure the selected fabric is available. For example, use `shm` only if all the processes can communicate with each other through shared memory. Use `dapl` only if all the processes can communicate with each other through a single DAPL provider. Ensure that the `dat.dll` library is in your `%PATH%`. Otherwise, use the `-genv` option for `mpiexec.exe` for setting the `I_MPI_DAT_LIBRARY` environment variable with the fully-qualified path to the `dat.dll` library.

---

6. Debugging and Testing

The Intel® MPI Library supports for the Microsoft* Visual Studio* debugger. Check the *Intel® MPI Library Release Notes for Windows* OS* for a full list of supported Microsoft Visual Studio versions. Check the Microsoft Visual Studio documentation for details on how to link with third party libraries.

In addition to using a debugger tool, this section explains some additional debugging capabilities of the Intel MPI Library.

6.1. Logging

Sometimes debugging an application is not effective and you prefer to use logging instead. There are several ways to get logging information from running applications.

6.1.1. Getting Debug Information

Environment variable `I_MPI_DEBUG` provides a very convenient way to get information from an MPI application at runtime. You can set value of this variable from 0 (the default value) to 1000. The higher the value, the more debug information you get.

```
> mpiexec.exe -genv I_MPI_DEBUG 5 -n 8 ./my_application
```

NOTE

High values of `I_MPI_DEBUG` can output a lot of information and significantly reduce performance of an application. A value of `I_MPI_DEBUG=5` is generally a good starting point, which provides sufficient information to find common errors. See the `I_MPI_DEBUG` description in Intel® MPI Library Reference Manual for Windows* OS for more details.

6.1.2. Tracing an Application

Use the `-t` or `-trace` option to link the resulting executable files against the Intel® Trace Collector library. This has the same effect as when `-profile=vt` is used as an argument to `mpiicc` or another compiler script.

```
> mpiicc -trace test.c -o testc
```

To use this option, you need to:

- Install the Intel® Trace Analyzer and Collector first. The tool is distributed as part of the Intel® Parallel Studio XE Cluster Edition bundle only.

- Include the installation path of the Intel® Trace Collector in the `VT_ROOT` environment variable. Set `I_MPI_TRACE_PROFILE` to the `<profile_name>` environment variable to specify another profiling library. For example, set `I_MPI_TRACE_PROFILE` to `vtfs` to link against the fail-safe version of the Intel® Trace Collector.

6.1.3. Checking Correctness

Use `-check_mpi` option to link the resulting executable file against the Intel® Trace Collector correctness checking library. This has the same effect as when `-profile=vtmc` is used as an argument to `mpiicc` or another compiler script.

```
> mpiicc -profile=vtmc test.c -o testc
```

Or

```
> mpiicc -check_mpi test.c -o testc
```

To use this option, you need to:

- Install the Intel® Trace Analyzer and Collector first. The tool is distributed as part of the Intel® Parallel Studio XE Cluster Edition bundle only.

- Include the installation path of the Intel® Trace Collector in the `VT_ROOT` environment variable. Set `I_MPI_CHECK_PROFILE` to the `<profile_name>` environment variable to specify another checking library.

For more information on the Intel® Trace Analyzer and Collector, see the documentation provided with this product.

6.1.4. Gathering Statistics

If you want to collect statistics on MPI functions used in your application, you can set the `I_MPI_STATS` environment variable to a number, between 1 to 10. This environment variable controls the amount of statistics information collected and the output to the log file. By default, no statistics are gathered.

For details, see Statistics Gathering Mode in *Intel® MPI Library Reference Manual for Windows* OS*.

6.2. Testing the Installation

To ensure that the Intel® MPI Library is installed and functioning correctly, complete the general testing according to the following instructions, in addition to compiling and running a test program.

To test the installation:

1. Verify through the Computer Management console that the Hydra service has been started. This calls for initialization of the Intel MPI Process Manager.

2. Verify that `<installdir>\intel64\bin` is in your path when running on Intel® 64 architecture:

   ```
   > echo %PATH%
   ```

   You should see the correct path for each node you test.

3. **(SDK only)** If you use the Intel® compilers, verify that the appropriate directories are included in the `PATH` and `LIB` environment variables:

   ```
   > mpiexec.exe -hosts 2 host1 1 host2 1 a.bat
   ```

   where `a.bat` contains

   ```
         echo %PATH%
   ```

You should see the correct directories for these path variables for each node you test. If not, call the appropriate `compilervars.bat` script. For example, when using the Intel® C++ Compiler 15.0 for Windows*OS for Intel® 64 architecture in 64-bit mode, use the Windows* OS program menu to select:
Intel(R) Parallel Studio XE 2015 > Command Prompt > Intel 64 Visual Studio 2012 mode or from the command line, and run

```
%ProgramFiles%\Intel\Composer XE\bin\iclvars.bat
```

6.2.1. Compiling and Running a Test Program

The install directory `<installdir>\test` contains test programs which you can use for testing. To compile one of them for your test program, do the following:

1. (SDK only) Compile a test program as described in Compiling and Linking.

2. If you are using InfiniBand* or other RDMA-capable network hardware and software, verify that everything is functioning.

3. Run the test program with all available configurations on your cluster.

4. Test the TCP/IP-capable network fabric using:

   ```
   > mpiexec.exe -n 2 -env I_MPI_DEBUG 2 -env I_MPI_FABRICS tcp
   ./myprog.exe
   ```

5. Test the shared-memory and DAPL-capable network fabrics using:

   ```
   > mpiexec.exe -n 2 -env I_MPI_DEBUG 2 -env I_MPI_FABRICS shm:dapl \
   ./myprog.exe
   ```

For each of the `mpiexec` commands used, you should see one line of output for each rank, as well as debug output indicating which fabric was used. The device(s) should agree with the `I_MPI_FABRICS` setting.

7. Process Management

This topic describes the process managers included with the Intel® MPI Library.

7.1. Scalable Process Management System (Hydra)

Hydra is a simplified, scalable process manager. Hydra will check for known resource managers to determine where the processes may be run and to distribute the processes among the targets using proxies on each host. These proxies will be used for process launching, cleanup, I/O forwarding, signal forwarding, and other tasks.

You can start Hydra by using `mpiexec.exe`. See Scalable Process Management System (Hydra) Commands topic for a detailed list of options in the *Intel® MPI Library Reference Manual*.

NOTE:

Multipurpose daemon* (SMPD) has been deprecated starting from Intel® MPI Library 5.0 release. To start parallel jobs, use the scalable process management system (Hydra).

7.1.1. Setting up Hydra Services

To run programs compiled with Microsoft* Visual Studio* (or related), first set up the `HYDRA` service.

NOTE

You should have administrator privileges to start the Hydra service. Once the Hydra service has started, all users can launch processes with `mpiexec.exe`.

The `Hydra` service is started during installation of the Intel® MPI Library. You can also select to cancel the Hydra service startup at this time.

You can start, restart, stop or remove the `Hydra` service manually when the Intel® MPI Library is installed. Those operations are done by executing `hydra_service.exe`, located in `<installdir>\intel64\bin`.

To set up `HYDRA` services:

1. Use the following command on each node of the cluster to remove the previous `Hydra` service: `hydra_service.exe -remove`.

2. Use the following command on each node of the cluster to install the `Hydra` service manually: `hydra_service.exe -install`.

7.2. Controlling MPI Process Placement

The `mpiexec` command controls how the ranks of the processes are allocated to the nodes of the cluster. By default, the `mpiexec` command uses group round-robin assignment, putting consecutive MPI process on all processor ranks of a node. This placement algorithm may not be the best choice for your application, particularly for clusters with symmetric multi-processor (SMP) nodes.

Suppose that the geometry is `<#ranks> = 4` and `<#nodes> = 2`, where adjacent pairs of ranks are assigned to each node (for example, for two-way SMP nodes). To see the cluster nodes, enter the command:

```
> type machines.Windows
```

The results should look as follows:

```
clusternode1

clusternode2
```

To equally distribute four processes of the application on two-way SMP clusters, enter the following command:

```
> mpiexec.exe -n 2 -host clusternode1 .\myprog.exe : -n 2 -host
clusternode2 .\myprog.exe
```

The output for the `myprog.exe` executable file may look as follows:

```
Hello world: rank 0 of 4 running on clusternode1

Hello world: rank 1 of 4 running on clusternode1

Hello world: rank 2 of 4 running on clusternode2

Hello world: rank 3 of 4 running on clusternode2
```

In general, if a cluster has `i` nodes and each node is a `j`-way SMP system, the `mpiexec` command-line syntax for distributing the `i*j` processes amongst the `i*j` processors within the cluster is:

```
> mpiexec.exe -n j -host <nodename-1> .\myprog.exe : \

-n j -host <nodename-2> .\myprog.exe : \

-n j -host <nodename-3> .\myprog.exe : \
...
-n j -host <nodename-i> .\myprog.exe
```

NOTE

You have to fill in appropriate host names from `<nodename-1>` through `<nodename-i>` with respect to your cluster system.

See Also

You can get more details in the *Local Options* topic of the *Intel® MPI Library Reference Manual* for Windows* OS.

You can get more information about controlling MPI process placement online at Controlling Process Placement with the Intel® MPI Library.

8. Tuning with mpitune Utility

This section describes how to use the `mpitune` utility to find optimal settings for the Intel® MPI Library.

8.1. Cluster-Specific Tuning

Intel® MPI Library has more than 100 parameters. The defaults are set for common usage and generally provide good performance for most clusters and most applications. However, if you want to get even higher performance, you can use the `mpitune` utility. This utility uses the Intel® MPI Benchmarks (IMB) as a benchmark program running tests several times with different parameters and searching for the best ones. Start the `mpitune` utility with the following command:

```
> mpitune
```

Then, start your application with the `-tune` option to enable the tuned settings:

```
> mpiexec.exe -tune -perhost 8 -n 64 ./myprog.exe
```

For best results, run `mpitune` with write access permissions for `<installdir>\<arch>\etc` since this is the default location for tuned parameters. If you do not have write access, a new configuration file will be saved in your current directory.

By default, `mpitune` uses the Intel® MPI Benchmarks (IMB) as benchmark program. Alternatively, you can substitute with your benchmark of choice by using the following command:

```
> mpitune -test \”your_benchmark –param1 –param2\”
```

You can then apply the new settings as described in Cluster-Specific Tuning.

The Intel® MPI Benchmarks executable files, which are more optimized for Intel microprocessors than for non-Intel microprocessors, are used by default. This may result in different tuning settings on Intel microprocessors than on non-Intel microprocessors.

8.2. Application-Specific Tuning

Use the `mpitune` utility to find optimal settings for your specific application.

```
> mpitune --application \”your_app\” --output-file yourapp.conf
```

Where `“your_app”` is the exact command line you use to start your application. For example:

```
> mpitune --application \".\myprog.exe\" --output-file $PWD\myprog.conf
```

Tuned settings will be saved in `myprog.conf` file. To apply them, simply call `mpiexec` as follows:

```
> mpiexec.exe -tune $PWD\myprog.conf -perhost 8 -n 64 .\myprog.exe
```

NOTE

`“myprog.exe”` is not only an executable file, but also a script which can be started as a separate process.

NOTE

The script should not change the `I_MPI_*` variables.

8.3. Setting Time Limit

The process of tuning can take a lot of time. Due to the varying factors involved in each cluster and application setup, the tuning time can be unpredictable.

To restrict the tuning time, you can set the `-time-limit` option. For example, to limit the tuning to 8 hours (480 minutes), run the following command:

```
> mpitune --time-limit 480
```

The time unit used is minutes.

8.4. Setting a Fabrics List

To define the fabrics to be tested, use the `fabrics-list` option.

```
> mpitune --fabrics-list shm,dapl
```

The available fabrics are: `shm:dapl`, `shm:tcp`, `shm`, `tcp`.

8.5. Setting a Range for the Number of Processes

To limit the number of processes running on one node, you can use the `perhost-range min:max` option. For example: the following command defines the number of MPI ranks on each node, between 4 and 8:

```
> mpitune --perhost-range 4:8
```

8.6. Setting a Limitation for Hosts Usage

To limit the number of nodes on which tuning will be performed, use the `--host-range min:max` option. For example, the following command will restrict running on 8 to 16 nodes only:

```
> mpitune --host-range 8:16
```

8.7. Restoring mpitune from the Last Saved Session

Sometimes an unexpected event can occur during the mpitune. In this case, you can use the intermediate saved in a `mpituner_session_<session-id>.mts` file. To restart `mpitune` from the last saved session:

```
> mpitune --session-file .\mpituner_session_<session-id>.mts
```

Where `<session-id>` is the timestamp of the moment tuner started.

8.8. Tuning Applications Manually

There is a family of `I_MPI_ADJUST_*` environment variables that allow you to manually tune the collective operations of the Intel® MPI Library. By setting a range of message sizes and choosing different algorithms, you can improve the performance of your application. For more information, see the `I_MPI_ADJUST` Family topic in *Intel® MPI Library Reference Manual for Windows* OS* for details.

9. Job Scheduler Support

The Intel® MPI Library supports the majority of commonly used job schedulers in the HPC field.

The following job schedulers are supported on Windows* OS:

- Microsoft* HPC Pack*
- Altair* PBS Pro*

9.1. Microsoft* HPC Pack*

The Intel® MPI Library job startup command `mpiexec` can be called out of Microsoft* HPC Job Scheduler to execute MPI application. In this case the `mpiexec` command automatically inherits the host list, process count and the working directory allocated to the job.

Use the following command to submit an MPI job:

```
job submit /numprocessors:4 /stdout:test.out  mpiexec -delegate
test.exe
```

Make sure the `mpiexec` and dynamic libraries are available through `PATH`. The Intel MPI Library environment variables can be registered during the installation process.

9.2. Altair* PBS Pro*

The Intel® MPI Library job startup command `mpiexec` can be called out of PBS Pro* job scheduler to execute an MPI application. In this case the `mpiexec` command automatically inherits the host list, process count allocated to the job if they were not specified manually by the user. The `mpiexec` reads `%PBS_NODEFILE%` environment variable to count a number of processes and use it as `machinefile`.

Example:

Content of job script:

```
REM PBS -l nodes=4:ppn=2

REM PBS -l walltime=1:00:00

cd %PBS_O_WORKDIR%

mpiexec test.exe
```

Use the following command to submit the job:

```
qsub -C "REM PBS" job
```

`mpiexec` will run two processes on each of four nodes for this job.

10. General Cluster Considerations

This topic discusses general considerations for clusters related to MPI usage:

- Defining which nodes to use
- Heterogeneous system and jobs
- User Authorization

10.1. Defining which Nodes to Use

By default, Intel® MPI Library looks for a file called `mpd.hosts`. This file should contain a list of all available nodes on the cluster which can be used for your application. The format of the `mpd.hosts` file is a list of node names, one name per line. Blank lines and the portions of any lines that follow a # character are ignored.

You can specify the full path to this file by using the `-f` option.

When running under a supported job scheduler, using the `-f` option is unnecessary as the hosts will be determined by the scheduler.

10.2. Heterogeneous Systems and Jobs

All clusters are not homogeneous. All jobs are not homogeneous. The Intel® MPI Library is able to run multiple sets of commands and arguments in one command line through two different methods.

The easiest option for running multiple commands in two methods is by creating a configuration file and defining the `-configfile` option. A configuration file contains a set of arguments to `mpiexec`, one group per line.

```
-n 1 -host node1 ./io <io_args>

-n 4 -host node2 ./compute <compute_args_1>

-n 4 -host node3 ./compute <compute_args_2>
```

Alternatively, a set of options can be passed on the command line by separating each group with ":".

```
> mpiexec.exe -n 1 -host node1 ./io <io_args> : -n 4 -host node2
./compute <compute_args_1> : -n 4 -host node3 ./compute
<compute_args_2>
```

When a process is launched, the working directory will be set to the working directory of the machine where the job was launched. To change this, use the `-wdir <path>`.

Use `-env <var> <value>` to set an environment variable to a value for only one process group. Using `-genv` instead will apply the environment variable to all process groups. By default, all environment variables are propagated from the environment at launch.

10.3. User Authorization

The Intel® MPI Library supports two main methods to allow user authentication across a Windows* OS cluster:

- Password-based authorization
- Domain-based authorization

The password-based authorization is the most common method of providing remote node access through a user's existing account name and password. Intel MPI Library allows users to encrypt their login information to the registry by the `mpiexec –register` or `wmpiregister` tools. This needs to be done once, during the first application run for the user.

The domain-based authorization uses Microsoft* Security Service Provided Interface* (SSPI*) to allow user authentication on the remote machines. This is done in accordance to the established domain policies as defined by the cluster administrator. No user account information (user name and password) is stored on the machine.

To enable the domain-based authorization method on a Windows* OS cluster, the administrator needs to setup Active Directory* delegation for the compute nodes and cluster users.

For full details on how to enable the Active Directory setup, see the User Authorization section of the *Intel® MPI Library Reference Manual for Windows* OS*.